HOST-BASED SYSTEMIC NETWORK OBFUSCATION SYSTEM FOR WINDOWS

THESIS

Kevin E. Huber, Civ

AFIT/GCO/ENG/11-05

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCO/ENG/11-05

Host-Based Systemic Network Obfuscation System for Windows

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Kevin E. Huber, B.S. Management Information Systems

June 2011

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT/GCO/ENG/11-05

HOST-BASED SYSTEMIC NETWORK OBFUSCATION SYSTEM FOR WINDOWS

Kevin E. Huber, B.S. Management Information Systems

Approved:

_____   _____
Dr. Barry E. Mullins (Chairman)   Date

_____   _____
Dr. Rusty O. Baldwin (Member)   Date

_____   _____
Mr. William B. Kimball (Member)   Date

AFIT/GCO/ENG/11-05

Abstract

Computer network traffic, specifically from Windows-based computers, can be used to identify a host's operating system and services. Operating system identification dramatically increases the effectiveness a computer attack. A host's operating system can be determined by viewing protocol headers and payloads. To effectively obfuscate the operating system, an obfuscation program must hide the operating system from multiple techniques. Current obfuscation programs have two limitations: the number of protocols obfuscated and the operating system the program can obfuscate. Most current programs obfuscate only the Linux operating system. These programs only obfuscate the TCP, UDP, and IP protocols and do not provide a complete obfuscation approach.

The Systemic Network Obfuscation System (SNOS) program obfuscates all protocol OSI layers for the Windows operating system. Nmap and Nessus test the obfuscation effectiveness of SNOS. Four separate hosts are used to test SNOS – Exchange server, SharePoint server, web server and workstation. SNOS' obfuscation effectiveness is compared to a benchmark and another Windows obfuscation program – OSfuscate. A network latency experiment determines the additional network latency induced by SNOS. SNOS increased network latency for two of the four hosts.

The Systemic Network Obfuscation Program successfully obfuscated the Windows operating systems against the techniques utilized by Nmap and Nessus and provides an effective obfuscation process throughout all the protocol layers of a network packet.

Acknowledgments

I am extremely appreciative for the time and commitment my family gave me during this thesis research. My wife was not only supportive but a key contributor in helping me to think clearly and adequately plan during my thesis research. My wife and oldest daughter spent hours reviewing and editing my thesis. My family also helped by keeping me grounded and lessening the stresses of thesis work through their laughter, games and love.

Special thanks to my advisor, Dr. Barry Mullins, for his guidance and help in choosing my thesis research and inspiring me to expand my knowledge and expectations. My committee also provided valuable insight into programming specifics within the Windows driver environment and analytical review of the results produced from the experiments. This thesis research was a culmination of the efforts and support of several people in helping me expand my knowledge of computers, networking, and the analytical process of conducting experiments.

Table of Contents

List of Figures

x

xi

List of Tables

Systemic Network Obfuscation System

I. Introduction

1.1   Research Motivation

Network packets can reveal the operating system and running services of a host. Ron Gula – the CEO of Nessus, a well-known vulnerability and exploitation analysis program, stated, "The ability to accurately classify an OS [operating system] is vital for automatic asset discovery and classification" [Gul09]. Without the ability to accurately identify the host operating system, an attacker would not be able to accurately craft a custom-tailored exploit because exploits depend upon the operating system of the target host.

The desire to hide the operating system from a possible attacker directed this research to explore the possibilities of obfuscating network packets. Previous research and programs focused on Linux-based operating systems; limited research is devoted primarily for the Windows operating system family. This research focuses on obfuscating the Windows operating systems, specifically Windows XP and Windows Server 2003, from known fingerprinting techniques.

1.2   Goals

The Windows operating system can be obfuscated by modifying the packets directly on the host. The goal of the Systemic Network Obfuscation System (SNOS) is to provide a complete set of obfuscation techniques for a Windows host to defeat

fingerprinting methods. To accurately obfuscate the Windows operating system on a host, SNOS obfuscates a network packet at every layer of the TCP/IP and OSI models which is discussed further in Section 4.1.2. SNOS defeats several fingerprinting techniques because fingerprinting programs – specifically Nmap and Nessus – utilize multiple methods to identify the host's operating system. In order to defeat these programs, the obfuscation method has to defeat each fingerprinting method.

1.3   Assumptions

Network packets are created and used by thousands of different protocols and services. The goal of SNOS is to obfuscate all layers of the TCP/IP model for a network packet. Application layer protocols represent a seemingly endless set of protocols and services that might need obfuscation; this research focuses on a subset of all available protocols and services.

The protocols and services are selected based on studies identifying the most common protocols and services on a network and the protocols used during fingerprinting techniques. The SNOS program only obfuscates protocols and services identified as a common protocol found on a network or a commonly used protocol for fingerprinting the host's operating system.

1.4   Previous Obfuscation Approach Limitations

Previous research focused on network-based devices to obfuscate network packets or focused on obfuscating network packets from a Linux operating system [Ber03] [RoS01]. Traffic normalization and transport scrubbing, two similar approaches, attempt

2

to eliminate protocol variances between operating system implementations. Although not specifically designed for operating system obfuscation, these two approaches can remove certain header fields that identify the Windows operating system. A proxy server approach intercepts packets and crafts a new normalized packet and sends that to the correct destination and is similar to a transport scrubber. These three similar methods were developed to work only on a Linux operating system. To achieve obfuscation for hosts running a Windows operating system, these methods can obfuscate an entire network of packets, like a firewall, by intercepting all network traffic before the packets leave the network.

None of these methods are capable of obfuscating all network traffic from a Windows operating system. These methods are implemented using network-based devices and therefore are designed to obfuscate network traffic leaving or arriving into the network. These methods cannot obfuscate a host's network packets transmitted within the network. OSfuscate is the only Windows-based program that claims to obfuscate the Windows operating system by running directly on the Windows operating system. OSfuscate focuses on a specific fingerprinting technique and does not provide a system wide obfuscation approach to defeat multiple fingerprinting techniques.

Section 2.5 provides additional details about each of these related obfuscation techniques, the limitations of each and is sub-divided into network-based and host-based obfuscation approaches. The Systemic Network Obfuscation System (SNOS) differs from these research topics by providing a Windows-based obfuscation program able to obfuscate all network packets from a Windows host. SNOS obfuscates all the OSI layers of a network packet to provide complete obfuscation process.

1.5   Research Overview

This research identifies the fingerprinting techniques and how these techniques can be defeated without losing functionality. The experimental process is divided into two separate experiments – obfuscation effectiveness which measures the success of SNOS in defeating fingerprinting techniques and network latency which compares the network performance degradation caused while running SNOS. For each experiment a benchmark trial is run to provide a base to compare against. For the obfuscation effectiveness experiment, OSfuscate runs as a third trial to test the effectiveness of SNOS relative to the results of OSfuscate against the fingerprinting programs. Chapter 3 further identifies the experimental design and Chapter 4 identifies the details and selection process of the SNOS program. Nmap and Nessus are used to test the obfuscation effectiveness of each trial. These two programs are selected because each uses multiple fingerprinting techniques.

The experiments are run within a virtual environment using the VMware ESXi hypervisor. Virtualization allows for repetition of each experiment while limiting the variable differences between each trial. The only variable difference between each trial is whether or not an obfuscation program is running on the host with three possible choices – no obfuscation (benchmark), OSfuscate, and SNOS.

Chapter 5 provides an observational and interpretive analysis of the results from both experiments. The obfuscation effectiveness results show a clear difference between SNOS and the other two trials – benchmark and OSfuscate.

II. Background

The following sections give an overview of how and why information must be protected, an overview of fingerprinting, the role of fingerprinting in network attacks, tools used for fingerprinting, and current tools and techniques available to minimize the effectiveness of fingerprinting.

2.1   Protecting Information

"Provided the enemy … is capable of reacting to what he sees, or thinks he sees, he can apparently be taken in again and again" [Bar52].

The primary role of most system/network administrators, technicians, or engineers is to protect information. Traditionally, important financial or other sensitive corporate or government information has been the focus of information protection. Computers and networks configured to protect sensitive information divulge information about themselves. Information pertaining to the end device can be used by someone with malicious intent to perform malicious activity and allow an attacker to gain control over that device, which ultimately leads to the compromise of information.

Important host information is easily accessible from host devices. Similar to a chatty employee around the water cooler, an attacker need only listen in on the conversation between employees or simply initiate the conversation with an unsuspecting employee [Kol05]. Modern operating systems and services either willfully or indirectly identify themselves in how they reply to various network traffic requests.

"A quick packet capture, on any network will show you that machines are chatty and will provide someone 'new' (a computer they know nothing about) info about their

OS, name, IP, MAC, locally logged on user, etc. Each OS has its own chatty nature and protocols it uses, some have been cleaned up, a bit, over the years, others don't appear to have been at all, and yet other protocols are being added that are designed to make life easier, but have added even more noise to the line" [Kol05].

Protecting sensitive host information, like the operating system, is often disparagingly referred to as 'security by obscurity'. Protecting information through obfuscation, masking or hiding sensitive information, is a well utilized approach. The simple act of leaving lights on in the house to appear that someone is home provides a simple example to the commonplace use and benefits attributed to the 'security by obscurity' defense mechanism [Yui06]. The military actively uses this approach by wearing camouflage [Mur09].

Modern firewalls and proxy servers hide information – such as the actual Internet Protocol (IP) address of an end device by using dynamic Network Address Translation (NAT) – at the network level. These techniques are an important step to protecting information by hiding sensitive information from an attacker. Despite the use of NAT and Access Control Lists (ACL), neither firewalls nor proxy servers currently protect the chatty sensitive information that most devices readily provide. The importance of obfuscating this sensitive host information has typically been downplayed and often ignored. Though obfuscation does exist within the information technology realm, typically obfuscation only plays an implicit role instead of an explicit one [Yui06]. Obfuscation and deception consists of determining the information someone should and should not know, control the focus, and prevent any undesired information from being observed [Rep08].

6

Host-based obfuscation increases adversary uncertainty and can increase the time and effort required to gain insight into possible attack vectors of the host. Many methods to obfuscate and conceal various aspects of the host device's identification have been studied and researched including normalizers, format alternation, scrambling identifiers, reordering content, and dynamic infrastructures [Rep08] and are discussed in Section 2.5.

2.2   Scanning / Planning

Some networking professionals estimate that an adversary spends up to 95% of their time preparing for an attack while only spending around 5% actually executing the attack [KFL01]. The preparation time is referred to as the planning phase in this research. An adversary is vulnerable during the planning phase because an attacker is, in some form, generating traffic on the network. During the planning phase, an attacker is attempting to glean as much information as possible from the network infrastructure.

Obtaining host device information is commonly referred to as fingerprinting. Several applications exist that allow someone to easily scan a network and determine the host operating systems and services running on each host system. Fingerprinting itself is not the immediate threat; fingerprinting is instead a precursor to an attack [WSM04]. Fingerprinting provides an attacker with sensitive information needed to specially craft exploits against the target host [Ark01]. Particularly, determining the host operating system is necessary for an attacker to correctly carry out a targeted exploit by identifying entry points, payloads, and existing vulnerabilities of the host operating system [Ber03] [Mur09]. Furthermore, knowing the host operating system could even allow an attacker

7

to set up a simulated environment similar to the target to find additional vulnerabilities [SkL08].

Figure 1 shows a Wireshark capture between two hosts located on the same local network. The figure shows two separate Browser packets – a Windows-generated packet on the right and a Linux-generated packet on the left.


```
Microsoft Windows Browser Protocol     |⊟ Microsoft Windows Browser Protocol
  Command: Host Announcement (0x01)     |    Command: Local Master Announcement (0x0f)
  Update Count: 0                        |    Update Count: 91
  Update Periodicity: 12 minutes        |    Update Periodicity: 12 minutes
  Host Name: 116-AMO-10                  |    Host Name: ANT
  OS Major Version: 5                    |    OS Major Version: 4
  OS Minor Version: 1                    |    OS Minor Version: 9
⊞ Server Type: 0x00001003               |  ⊞ Server Type: 0x00049a03
  Browser Protocol Major Version: 15     |    Browser Protocol Major Version: 15
  Browser Protocol Minor Version: 1      |    Browser Protocol Minor Version: 1
  Signature: 0xaa55                      |    Signature: 0xaa55
  Host Comment:                          |    Host Comment: Samba 3.0.10
```

Figure 1: Browser Protocol Wireshark Capture [Kol05]

The packets in Figure 1 indicate the OS Major and OS Minor Version. Although the number after these fields might not appear to be valuable, an attacker can simply look up the corresponding operating system in a database full of predefined header field patterns. "OS Major Version: 5" represents Windows and "OS Minor Version: 1" means that the version of Windows is XP. By quickly examining this one packet, an attacker now knows that the host computer is running Windows XP. The packet on the right of Figure 1 gives additional information located in the Host Comment field identifying the application and exact version number responsible for the service.

During this phase an attacker typically scans sections of a network to determine each host's operating system and the services running on each host. Several application-level protocols, like HTTP and SMTP, as well as TCP and IP header fields, are used to

identify the specifics of a host's operating system and services. The differences between the results are often easily discernable and are detailed in Section 2.3.

2.2.1   The Deception Process

Obfuscation is a type of deception. The deception process must produce results that continue to deceive irrespective of the attack method. To deceive an attacker, not only must false information be presented but any other valid information must remain hidden [Yui06]. This false information includes crafting packets to appear to have been created from different operating systems and applications.

To date, current attempts to obfuscate host-level information have only narrowly focused on certain aspects of host-level detection. None of the researched and well-known host-level obfuscation tools discussed in Section 2.5.2 provide a thorough defense against the many fingerprinting methods. A solid deception framework must be able to conceal information at multiple levels and from different fingerprinting methods.

Figure 2 depicts a decision flowchart necessary for determining the varying facets an attacker can use to obtain the important host information. A fingerprinting method is deployed against a host. After the host had been engaged and the results returned, the attacker can make a decision to attempt a new fingerprinting method against the host or terminate the fingerprinting process. If an attacker is deceived during the first attempt, the attacker would deploy additional fingerprinting methods. Nmap and Nessus use this multi-technique approach to fingerprinting a host; therefore, a deception should be verifiable against multiple fingerprinting methods [Yui06].
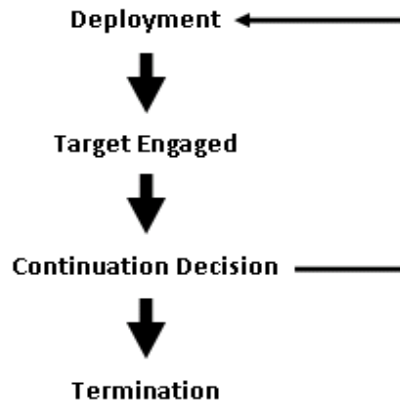
Figure 2: The Basic Deception Process [adapted from Yui06]

If the current obfuscation process only changes certain transmission control protocol (TCP) header values but the computer still advertises an Internet Information Service (IIS) service, then the TCP header obfuscation becomes meaningless. The IIS service implies that the host is running a Windows operating [Bec01].

2.3   Planning / Scanning Vectors

The three main sources an attacker can scan from are: external, malware, and insiders [Yui06]. Firewall solutions can stop external scanning against hosts behind the firewall. Firewall rule sets can include blocking unsolicited TCP sessions (initiations) and ICMP requests. Firewalls can be averted by performing different types of scans or implementing the scan differently. One possible way to bypass a firewall or IDS (intrusion detection system) is to simply change the maximum transmission unit (MTU) size or fragment the correlating packets in an attempt to exploit the differences between how a firewall de-fragments and fragments packets.

Malware located on an internal host is already inside the "trusted" local network, and has fewer security features to overcome in order to start scanning and identifying other local hosts. Computers located inside the "trusted" local network can become

infected through several different scenarios, ranging from social engineering attacks to man-in-the-middle attacks. Social engineering attacks are now fairly simple to orchestrate thanks to the wide range of script kiddie tools, so called because of their ease of use.

Once malware gains access on a computer, the malware can start locating other possible targets within the local network infrastructure. The Sapphire worm is a scan-and-attack example, once the computer is infected, the worm sends out UDP packets throughout the local network and beyond [Naz04].

An insider could be someone that is willingly, or just unknowingly, helping an attacker. The attacker could be the insider. Similar to the malware approach above, an insider has already bypassed network-based security features – such as the firewall or IDS and has more access to obtain host-level information from the remaining local hosts.

Mazu Networks, determined that 23% of organizations in the United States with more than 1,000 employees had at least one internal security breach during 2004 with another 27% not sure if a compromise was internal or external [Yua05]. The survey shows the fundamental problem of simply relying on a network-based firewall or IDS in an effort to protect host-level information.

2.3.1 Fingerprinting Techniques

Host-level information can also be extracted from network traffic by sifting through network packet headers and pattern similarities. Similar to how law enforcement utilizes fingerprints, computer fingerprinting techniques obtain a set of patterns by scanning a host and viewing the network traffic from that host. The computer fingerprint

11

is compared with a list, or database, of various network traffic patterns. Software vendors, including the operating system developers, implement network protocols differently.

The different implementations of a protocol stack are allowed because protocol specifications leave ambiguities while implementing optional fields and the order and value of specific header fields. The ambiguities within protocol specifications allow an attacker to view the characteristics of how the target host implements a specific protocol and matches the results with the known characteristics of a specific operating system or service. Protocol ambiguities allow for more sophisticated scanning [WSM04].

Two general probing techniques are commonly deployed to determine host information: active and passive fingerprint probing. Active probing involves sending a custom crafted packet to the target host and listening for the reply. Therefore, active probing requires both the target host and the attacker to participate. These specially crafted packets include three common IP packet types (ICMP, TCP, and UDP) and are crafted to use both standard and non-standard protocol implementations [KaS10]. Active probing, if coming from an external source, can be blocked by a firewall or IDS [BHP07]. The attacker uses the reply packet as the fingerprint of the target host. Active probing/scanning gives an attacker a more complete view of the host and services running on the host. Active probing can usually produce information about a host much faster than passive probing.

Passive probing is deployed by observing the normal network traffic already generated by the target host and is stealthier because it does not generate network traffic. Passive probing only sees traffic destined to itself or traffic sent as a broadcast or

multicast message unless the attacker can perform a man-in-the-middle attack or the network uses a hub or spanning port on a switch. Passive probing does not produce as much host information because the network enabled services on a host wait for a request before transmitting response packets. If a host does not get a request on a specific service port number during the time an attacker is passively probing, then the attacker will never see any traffic regarding that running service. Passive probing increases the time needed to scan a target for host-level information.

A lesser-utilized fingerprinting approach is aptly named exploit testing. This approach initiates a series of network traffic at a target with the attempt to create a denial-of-service attack specific to an operating system. If the target host crashes accordingly, then the attacker can potentially determine the host operating system given the specially crafted denial of service attack performed [Bec01]. This method of fingerprinting is both very noisy and provides a more limited set of host-level information than the two previously mentioned methods. In general, exploit testing is not commonly used to fingerprint a host, unless the primary goal of an attack is to simply crash the target host.

2.3.1.1 Fingerprinting Specifics Overview

Irrespective of the fingerprinting technique used, fingerprinting is more effective if the target host has multiple services running with significant amounts of information transmitted over the network that can be used to narrow down the fingerprint of the operating system and services [Fyo02].

Passive and active fingerprinting techniques inspect the implementation details of the target's protocol stack, including the application, TCP, and IP layers. Common

header fields like the time-to-live (TTL), window size, Don't Fragment bit (DF), type of

service (TOS), IP ID sequence number and TCP sequence numbers provide information

in determining the host's operating system [Mur09] [Fyo02]. Figure 3 shows some of the

default TTL values according to the specific operating and how these values infer the

host's operating system. The default TTL values are shown on the right for each

operating system; Windows XP's default TTL value is 128. An attacker can view the

TTL header field from an IP packet and identify the target's operating system.

SunOS 5.6	255
Ultrix 4.2-4.5	255
Ultrix 4.2-4.5	255
USR8000 broadband router	64
Web V Networks Web Cam	64
Windows 2000	128
Windows 2003	128
Windows 95 Original	32
Windows 95A or B w/winsock 2	128
Windows 98 or 98SE	128
Windows ME	128
Windows NT 4	128
Windows XP	128

Figure 3: Default OS-specific TTL values [Kol05]

2.3.2   Banner Grabbing

Another method of fingerprinting, which was a technique used by early hackers,

is referred to as banner grabbing [Ber03]. The banner refers to the general message the

host application displays to a user accessing that service. The banner is a string of text

with a general announcement that typically includes specifics on host services [SiB07].

These banner messages are common and still widely used today in applications such as

FTP, telnet, SSH, SMB, and SMTP. Sometimes the banner specifically displays the

operating system, the application version, and patch level. Other times, the banner

14

displays a correlated number referencing the operating system or application name and/or version number.

Even though banner grabbing is a well known means for obtaining sensitive host information, application developers and appliance vendors still display sensitive host information in the default banners. Even security related appliances and services sometimes provide this information. Figure 4 and Figure 5 are live banners currently viewable to anyone connecting to either of these hosts on the Internet.
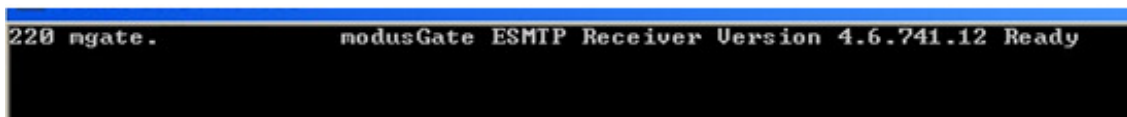
Figure 4: Email Filter Banner

Figure 4 shows a security device, in this case an email filtering appliance that displays both the application and the current installed and running version of the application.
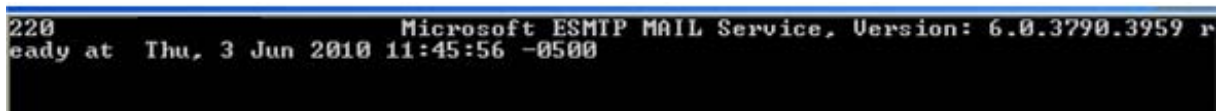
Figure 5: Microsoft Exchange Banner

Figure 5 provides shows a banner displaying Microsoft ESMTP which implies that the email server is a Microsoft Exchange server. Version 6.0 means the Exchange server is running Exchange Server 2003; because the server is running Exchange, the host operating system is a version of Windows server, most likely Windows Server 2003 [GLM10]. Banner grabbing not only references the default messages services transmit over the network, but also certain application-layer protocol-related header fields, such as the Server field in a HTTP header. Several of these fields are reviewed in more detail in the subsequent sections. A thorough obfuscation method must address banner grabbing.

15

Obfuscating the entire TCP/IP stack successfully is meaningless if an attacker can view

the banner message from the server application and then glean all the information the

TCP/IP stack obfuscation processes protects [Bec03].

2.3.3   TCP / IP Stack Fingerprinting

The IP header, Figure 6, contains some of the most common fields used to

identify a host operating system. The 3-bit Flags field in the IP header consists of the

Reserved bit (1st bit), the Don't Fragment (DF) bit (2nd bit), and the More Fragments bit

(3rd bit). Each one of these three bits can be used to identify the host, particularly the DF

bit, because operating systems utilize and respond differently depending on which bits

are set.
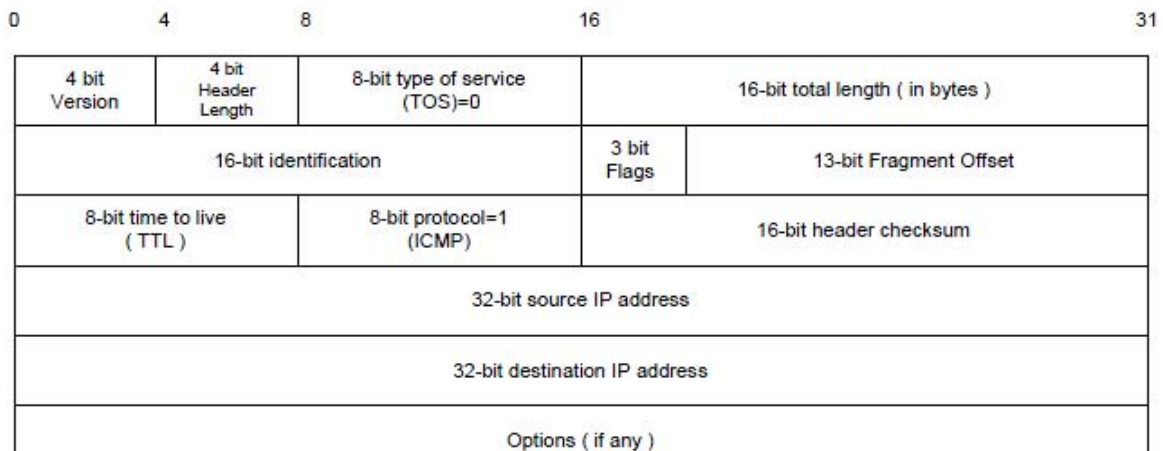
Figure 6: IP Packet [Ark01]

The first stack querying/fingerprinting methods targeted the IP and TCP protocols

[Spa03]. TCP fingerprinting sends both standard and malformed TCP packets to the

target host and then analyzes the responses [VCH02]. New service packs for an

operating system sometimes implements the TCP and IP protocols with slight variations.

Several fingerprinting programs, such as Nmap and Nessus, use TCP querying to

determine the host operating system (OS) and services on the target. The TTL example from Figure 3 demonstrates how some of the information from the host's response can be used to fingerprint the host's OS and services.

The TCP header field variations, in Figure 7, between different operating systems can imply the underlying operating system. The window size field and the sequence of any options within the Options field are useful in TCP fingerprinting. The flags portion of the TCP header consists of the Congestion Window Reduced (CWR) bit, the ECN-Echo (ECE) bit, the Urgent (URG) bit, the Acknowledgement (ACK) bit, the Push (PSH) bit, the Synchronize (SYN) bit, and the Finish (FIN) bit. The SYN bit is used by the first packet to synchronize the sequence numbers and the ACK bit is set when replying to all the packets initiated from SYN bit packet. These flag bits are used for operating system fingerprinting by setting seldom used or seldom grouped together bit fields to see how the target operating system responds.
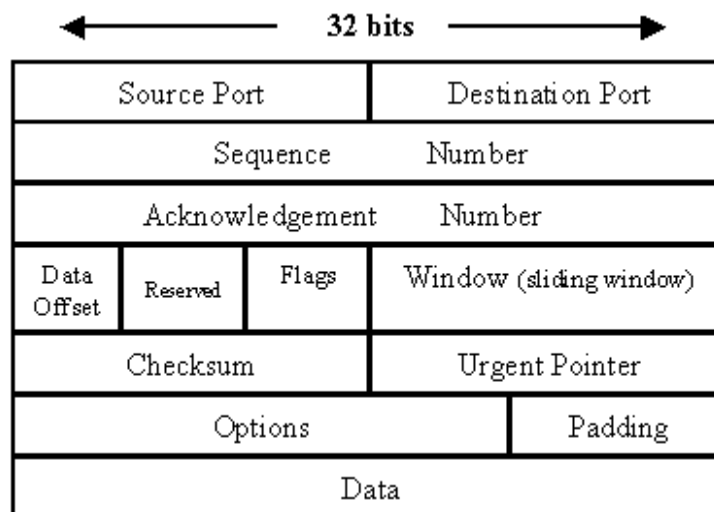
Figure 7: TCP Packet

One of the first steps to query a host's operating system is to set the synchronize bit to detect open ports and services. Operating systems utilize different ports. Windows

operating systems typically have port 139 open which is the NetBIOS session or Windows File and Printer Sharing default port. Some firewalls and IDSs are able to block these requests by not allowing an external device to initiate a connection with the target device. But certain applications, like a web or email server, have to allow initial TCP requests in from external sources and therefore the SYN technique can still target these hosts. Any SYN packet sent from within the LAN would also be successful as the scan is already beyond the reach of a network firewall. A thorough obfuscation program must also block unused ports that are opened by default to limit the amount of data correlation available from the scanned results of a host.

Another simple TCP fingerprinting method is to send a packet with the ACK bit set and observe the response from the host. Some operating systems drop the packet while other operating systems send back a TCP packet with the RST flag set. The ACK technique can bypass firewall rule sets, but a stateful firewall – a firewall that remembers TCP session states – should drop this packet. A network firewall is useful against external attack vectors but provides limited obfuscation benefits for internal network traffic. A similar approach is to send a TCP packet with the RST bit set and wait for the response back from the host. The RST method has similar advantages and disadvantages as the ACK method.

Another well utilized method of stack querying is the Initial Sequence Number (ISN) analysis. An operating system initials a random ISN to deter session hijacking; however, the range of the possible ISNs can be used to imply which operating system the host is currently running. Another fingerprinting method referred to as temporal response analysis observes the retransmission timeout (RTO) responses from a host to imply

18

operating system specifics [Spa03]. A more in-depth view of the different approaches to

TCP stack querying and current tools used to conduct TCP stack querying is detailed in

Section 2.4.

TCP fingerprinting can be done passively and actively. With active TCP

fingerprinting the attacker sends malformed packets to view responses from the target.

To more accurately identify a host, an attacker uses several TCP fingerprinting methods

at the same time, increasing the likeliness the results correlate with the same operating

system. TCP fingerprinting allows the attacker to gain a fairly comprehensive

understanding about the configuration of the host by viewing the open ports and

determining the services running on those ports.

2.3.4 ICMP Fingerprinting

Figure 8 shows how the Internet Control Message Protocol (ICMP) packet is

constructed. The possible values for the ICMP header fields provide the means for an

attacker to gain insight into the operating system running on the host machine.

Information is gleaned from the TOS, type of service, byte which consists of the

Precedence Bits, TOS bits, and the Unused bit. A standard ICMP packet will have the

TOS field set to zero; attackers populate the TOS bit to help fingerprint the target by

viewing how the target operating system responds.

| 0 | 4 | 8 | 16 | 31 |
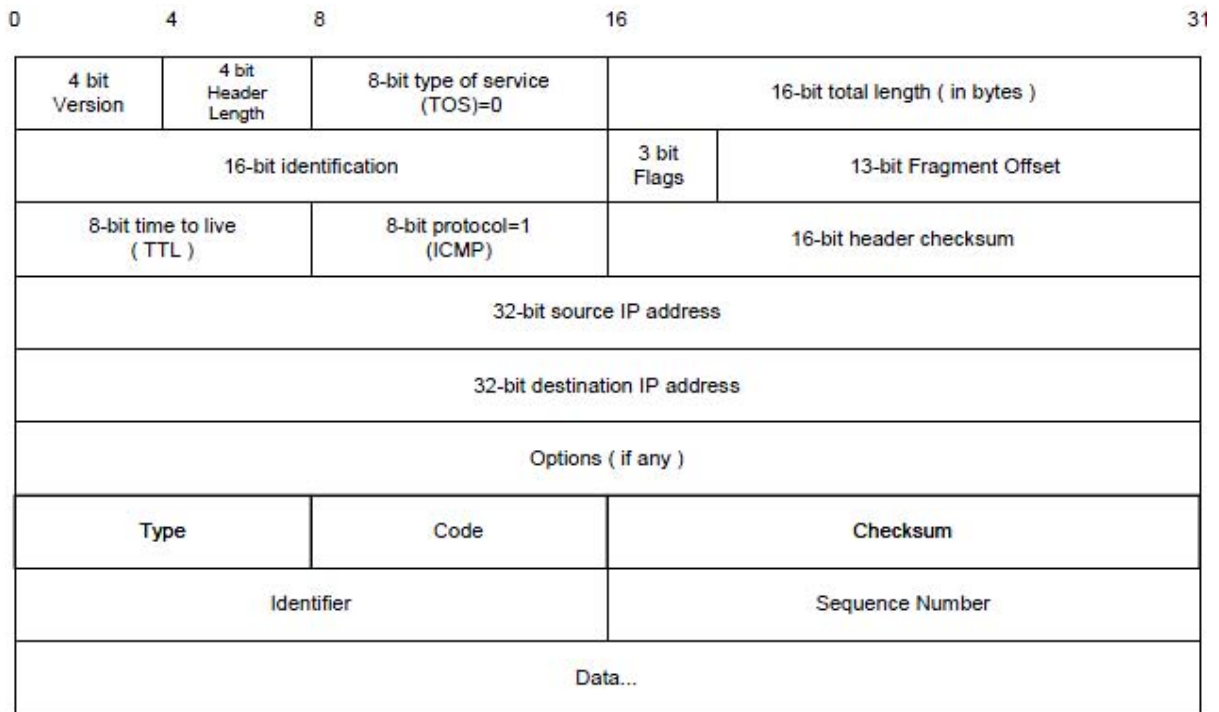
Figure 8: ICMP Echo Request Message Format [Ark01]

In Figure 8, the Type field in the ICMP header indicates the type of message being relayed. Some the possible types include: destination network unreachable, destination host unreachable, fragmentation required, echo reply, echo request, timestamp reply, address mask request, etc. The Type field allows for 255 different message types – although less than 50 are actively used. The Code field relates to the Type field and provides additional supporting information.

The ICMP identifier field has a constant value that can be used to identify the host operating system [Ark01]. Windows operating systems send packets with an Identifier of 0x0200, 0x0300, or 0x0400 [Kol05]. Figure 9 is a list of the common ICMP Identifier field values and the associated operating system.

```
0x4757 Novell Netware
0x1b04 ???
0x0000 Linux Kernel 2.4x (replies only)
0x0100 Unknown (abcd for data)
0x0100 Windows (rare?, possible NT4?)
0x0200 Windows (most common)
0x0300 Windows (2nd most common)
0x0400 Windows (3rd most common)
```
Figure 9: ICMP Identifier Field values [Kol05]

The ICMP Sequence number increments for each corresponding ICMP packet. For Windows operating systems, the sequence number increments by 0x0100 (256) for each proceeding ICMP packet and most Novel Netware operating systems keep the sequence number at 0x0000 for the duration of the current set of pings [Kol05].

For ICMP packets, the TTL value inside the IP header depends on if the ICMP message is a request or a response. As previously discussed in regards to TCP/IP stack fingerprinting and shown in Figure 3, the TTL IP header field is similarly useful in inferring the host's operating system for an ICMP message.

The data field in Figure 8 is the payload of the ICMP packet. Operating systems use different payloads which makes the fingerprinting process easier. Linux operating systems use a timestamp followed by random bits ending with 1234567. Windows operating systems use a 32 byte payload of repeating characters of the alphabet starting with 'A' [Kol05]. The size of the ICMP data field is also different between a Windows operating system and a Linux operating system [Ark01].

TOS Echoing is a specially-crafted ICMP packet that changes the value in the TOS IP header field. Another specially-crafted ICMP packet sets the 'Unused bit' in the IP header. Setting the DF (Don't Fragment) bit for an ICMP request produces varying responses from different operating systems. When sending ICMP query messages with

the Code field set to any value other than zero, Windows operating systems echo back the Code value that the host received while Unix-based hosts will set the value to zero in the response. Setting the ICMP Address Mask field is another technique to identify which version of Unix-based operating system the host is running [Ark01] because different versions of Unix-based operating systems populate this field uniquely.

All of the ICMP methods require observing the responses received from the target host. Using ICMP to fingerprinting a host, an attacker can determine if the host operating system generates ICMP Protocol Unreachable Error Messages, ICMP Error Messaging Quenching, ICMP Error Message Quoting Size, or ICMP Error Message Quoting Size Differences. The last error message is particular only for Linux-based operating systems. Several other error-related messages identify the host machine, such as: ICMP Error Message Echoing Integrity, or observing the Precedence, TOS, or DF bits from ICMP Error messages [Ark01].

The ICMP protocol allows for several types of ICMP messages. The first 8 bits, as shown in Figure 8, of the ICMP header are used to distinguish the type of ICMP message being transmitted. Operating systems do not implement every possible ICMP message type and therefore soliciting replies using different ICMP message types can also reveal the operating system running on the host. Some of these additional, lesser known ICMP message types are: ICMP Time Stamp Request and Reply and ICMP Information Request and Reply [Ark01].

The ICMP protocol receives a significant amount of attention at the network layer as network administrators often configure firewalls to block ICMP requests and replies from leaving the network. Local area network (LAN) traffic often allows ICMP

messages between hosts. Once an attacker has compromised a single host inside the LAN or if the attacker is an 'insider' then that attacker can carry out ICMP fingerprinting methods to identify additional targets on the local network.

2.3.5   DHCP Fingerprinting

Dynamic Host Configuration Protocol (DHCP) is used to dynamically configure a host's network settings to allow the host to use the network. These settings include the IP address, Subnet Mask, the default Gateway, and Dynamic Name Service (DNS) servers, as well as other configuration settings.

The DHCP process is shown in Figure 10. The first step shown in Figure 10 is when a new host on the LAN sends out a broadcast DHCP Discovery message in an attempt to locate a DHCP server. When the broadcast message reaches a DHCP server, the server replies back to the host with a DHCP Offer message. When the host receives the DHCP Offer packet, the host then sends a DHCP Request packet back to the DHCP server. The server responds by sending the host a DHCP ACK packet containing the LAN configuration settings. Figure 10 shows some of the possible deviations from this normal flow of DHCP messages for different possible scenarios – DHCP configuration's lease time has exceeded or another host already has the IP address assigned.
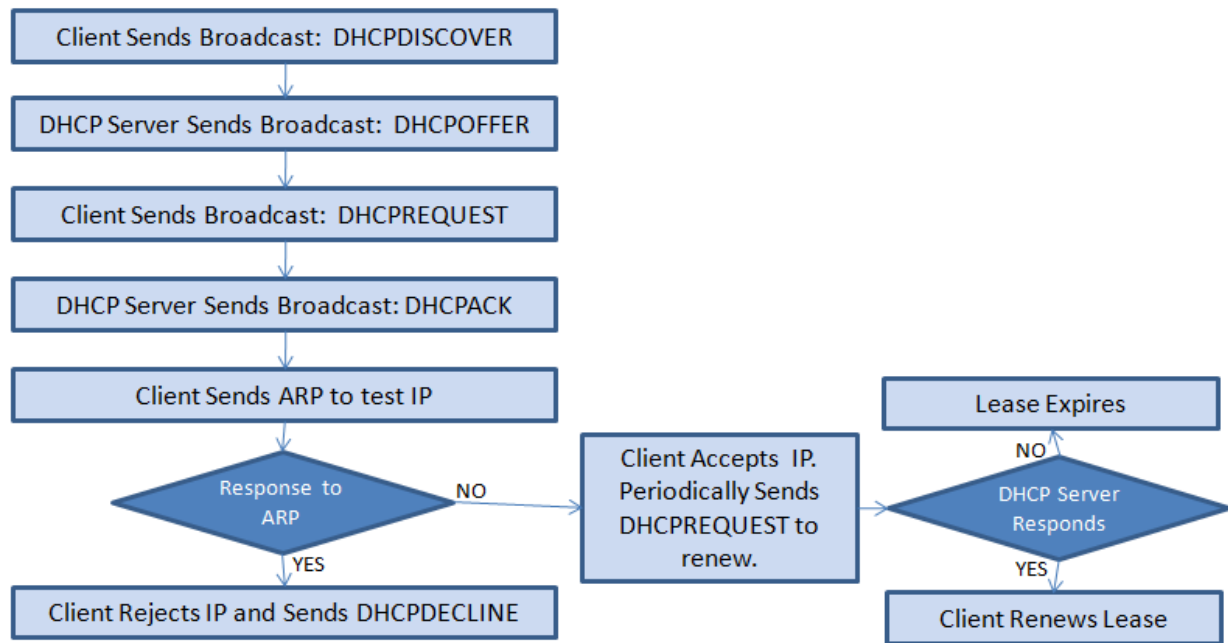
Figure 10: DHCP Flowchart [adapted from Kol07]

A host operating system requests specific options from a DHCP server. These options vary between operating systems. Figure 11 identifies some of the options and the parameters of a DHCP packet. The hostname is visible so organizations that use a service-oriented naming convention, i.e. calling a web server web1, allow an attacker to know open ports and services running on that host. Sometimes organizations include the operating system type in the hostname of the device, such as web1win2003 which identifies that the host is running Windows Server 2003.

Other important fields, shown in Figure 11, are the Differentiated Services Field, the Seconds elapsed field, and the Bootstrap Protocol (BOOTP) flags. The BOOTP flags are a legacy of BOOTP which DHCP replaced. These bits are typically all 0's. Some of these fields vary between operating systems. Option 55 is one of the most useful fields for DHCP fingerprinting because it is the Parameter Request List which indicates the

order of the DHCP options are being requested by the host [Kol07] and operating systems

order the options differently.

```
Bootstrap Protocol
   Message type: Boot Request (1)
   Hardware type: Ethernet
   Hardware address length: 6
   Hops: 0
   Transaction ID: 0xa000a000
   Seconds elapsed: 0
 ⊞ Bootp flags: 0x0000 (Unicast)
   Client IP address: 0.0.0.0 (0.0.0.0)
   Your (client) IP address: 0.0.0.0 (0.0.0.0)
   Next server IP address: 0.0.0.0 (0.0.0.0)
   Relay agent IP address: 0.0.0.0 (0.0.0.0)
   Client MAC address: Vmware_8b:69:93 (00:0c:29:8b:69:93)
   Server host name not given
   Boot file name not given
   Magic cookie: (OK)
 ⊞ Option: (t=53,l=1) DHCP Message Type = DHCP Discover
 ⊞ Option: (t=61,l=7) Client identifier
 ⊞ Option: (t=50,l=4) Requested IP Address = 192.168.0.27
 ⊞ Option: (t=12,l=5) Host Name = "w95b"
   End Option
```
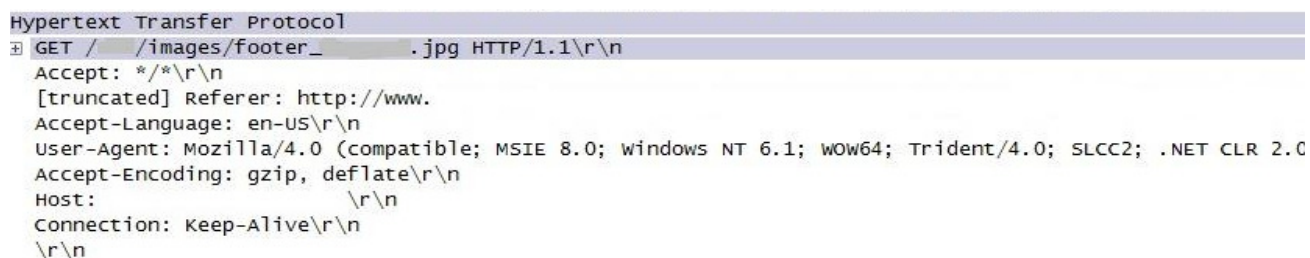
Figure 11: DHCP Packet [Kol07]

A complicated and less reliable DHCP fingerprinting method is to observe the

various time delays the host takes before trying to send out another DHCP Discovery

packet when a DHCP server is not initially found by the host [Kol07]. A host waits a

pre-determined amount of time, depending upon the host's operating system, before

retransmitting a DHCP Discovery packet.

Application fingerprinting observes the data transmitted from application layer

protocols. These application layer protocols include: Hyper-Text Transfer Protocol

(HTTP), Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP) and

Secure Shell (SSH).

2.3.6   HTTP Fingerprinting

The HTTP protocol can be divided into request and response messages. One of the header fields in a HTTP request message is the User-Agent field, shown in Figure 12 – which shows a Wireshark capture of an HTTP request message. The User-Agent field indicates the type and typically the version of the web browser that the host is running. The User-Agent field in Figure 12 indicates that the host's web browser is Microsoft Internet Explorer version 8.0. Also, inside the User-Agent field the text ".NET CLR 2.0" means that the host is running Microsoft .Net version 2.0 [Kol05] which can provide further avenues of attack.

```
Hypertext Transfer Protocol
⊞ GET /     /images/footer_          .jpg HTTP/1.1\r\n
  Accept: */*\r\n
  [truncated] Referer: http://www.
  Accept-Language: en-US\r\n
  User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0
  Accept-Encoding: gzip, deflate\r\n
  Host:                    \r\n
  Connection: Keep-Alive\r\n
  \r\n
```
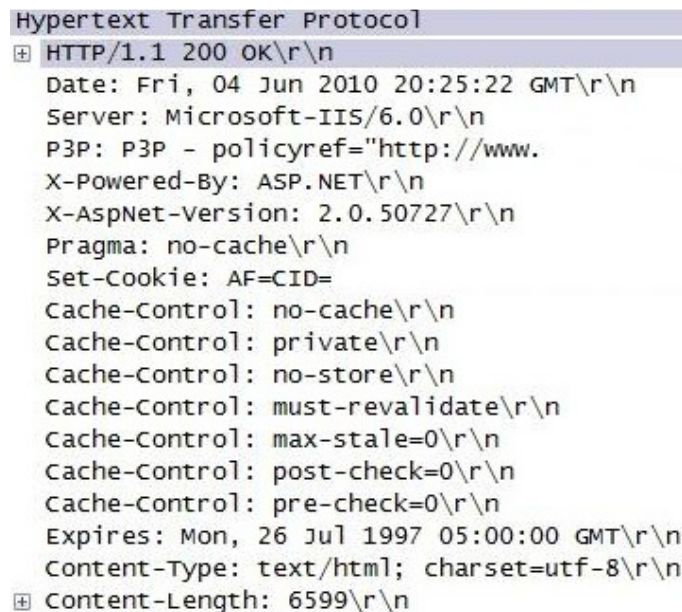
Figure 12: HTTP Request Packet

The User-Agent HTTP header field is often used by web developers to determine the Hyper-Text Markup Language (HTML), JavaScript, and Cascading Style Sheet (CSS) implementations that are needed by the requesting host. Web developers rely on this field in order to respond with the correct version of the web page that is best suited for a particular web browser's implementation. The variance in web browser functions, features, and how items are displayed – using HTML, JavaScript, and CSS – are a major nuance for web developers. Web browsers add new variations to the web browser's content markup and scripting while ignoring portions of the standards set forth. Since

web browsers depend on portions of this information, the functionality of the web browser is tested with SNOS running.

An attacker must either passively monitor all traffic from a host or perform some kind of DNS poisoning to the host's DNS servers so that the host's HTTP Request message would be sent to the attacker's machine or use some form of social engineering. An HTTP Response message is the response to an HTTP Request. HTTP Response packet fingerprinting can be performed by actively sending an HTTP Request message to the host and waiting for the response.

Figure 13 illustrates a typical HTTP Response message back from a web server. In Figure 12 and 13, IP addresses, domain names, and other identifying information have been removed because these packets were captured from live servers on the Internet.

```
Hypertext Transfer Protocol
⊞ HTTP/1.1 200 OK\r\n
   Date: Fri, 04 Jun 2010 20:25:22 GMT\r\n
   Server: Microsoft-IIS/6.0\r\n
   P3P: P3P - policyref="http://www.
   X-Powered-By: ASP.NET\r\n
   X-AspNet-Version: 2.0.50727\r\n
   Pragma: no-cache\r\n
   Set-Cookie: AF=CID=
   Cache-Control: no-cache\r\n
   Cache-Control: private\r\n
   Cache-Control: no-store\r\n
   Cache-Control: must-revalidate\r\n
   Cache-Control: max-stale=0\r\n
   Cache-Control: post-check=0\r\n
   Cache-Control: pre-check=0\r\n
   Expires: Mon, 26 Jul 1997 05:00:00 GMT\r\n
   Content-Type: text/html; charset=utf-8\r\n
⊞ Content-Length: 6599\r\n
```
Figure 13: HTTP Response Packet

A useful field for fingerprinting the HTTP Response message is the Server field. The Server field displays the type of web server running – a Microsoft Internet Information Services (IIS) version 6.0. Based on the type of web server, an attacker can

27

infer the operating system of the host [Fyo02]. Sometimes the Server field actually displays the operating system in parenthesis immediately after the type of web server version. The X-Powered-By field also indirectly indicates the type and version number of the web server and therefore the operating system.

Similar to TCP fingerprinting, HTTP fingerprinting involves sending non-standard HTTP Request packets to the web server, such as: sending a delete instead of a get, improperly identifying the HTTP version – e.g., HTTP/3.0, or a response with an improper protocol specified – e.g., Junk/1.0 [Sha04]. The order of the HTTP Response header fields vary depending upon the web server implementation, evidenced in Figure 14 below which shows two HTTP Response messages from different web servers running on different operating systems. These web servers, an Apache/1.3.23 and a Microsoft IIS/5.0, order the HTTP Response header fields differently. The message field changes when non-standard HTTP packets are sent to a web server. Some web servers ignore the non-standard parts and still serve up a valid HTTP Response while other web servers respond with an error message, e.g. HTTP/1.1 400 Bad Request [Sha04].

From an Apache 1.3.23 server:

```
HTTP/1.1 200 OK
Date: Sun, 15 Jun 2003 17:10:49 GMT
Server: Apache/1.3.23
Last-Modified: Thu, 27 Feb 2003 03:4
ETag: "32417-c4-3e5d8a83"
Accept-Ranges: bytes
Content-Length: 196
Connection: close
Content-Type: text/html
```

From a Microsoft IIS 5.0 server:
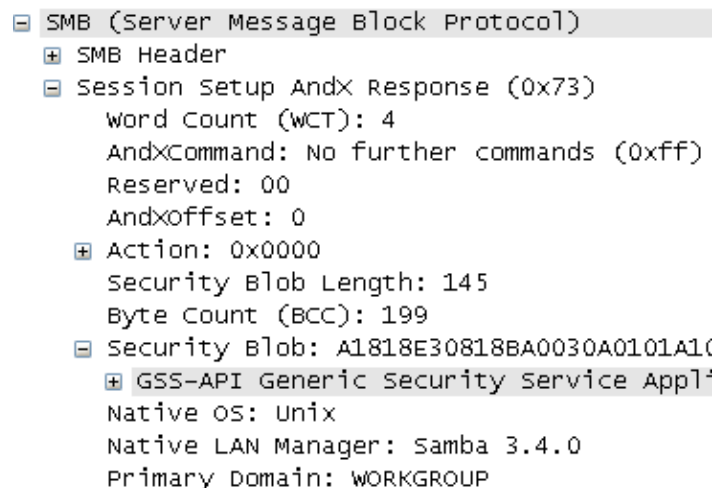
```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Expires: Tue, 17 Jun 2003 0
Date: Mon, 16 Jun 2003 01:4
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Wed, 28 May
ETag: "b0aac0542e25c31:89d"
Content-Length: 7369
```

Figure 14: Web Server Dependant HTTP Response Messages [Sha04]

2.3.7 SMB Fingerprinting

Server Message Block (SMB) fingerprinting is another avenue to obtain host information being transmitted in a network packet. SMB is an application-layer protocol like HTTP, SMTP, and FTP and is primarily used to provide shared access to files and printers.

SMB uses several subcommand layers. Figure 15 shows a Session Setup AndX Response subcommand packet which provides the most obvious OS identification fields. The Native OS and Native LAN Manager fields identify the operating system and the service and version using the SMB protocol.

```
⊟ SMB (Server Message Block Protocol)
   ⊞ SMB Header
   ⊟ Session Setup AndX Response (0x73)
        Word Count (WCT): 4
        AndXCommand: No further commands (0xff)
        Reserved: 00
        AndXOffset: 0
      ⊞ Action: 0x0000
        Security Blob Length: 145
        Byte Count (BCC): 199
      ⊟ Security Blob: A1818E30818BA0030A0101A10
        ⊞ GSS-API Generic Security Service Appli
        Native OS: Unix
        Native LAN Manager: Samba 3.4.0
        Primary Domain: WORKGROUP
```
Figure 15: SMB Protocol

2.4 Current Fingerprinting Tools

2.4.1 Nmap

Nmap is one of the most respected and popular fingerprinting tools freely available on the Internet and runs on most operating systems. Nmap has a large database backend to identify thousands of different host operating systems by comparing results
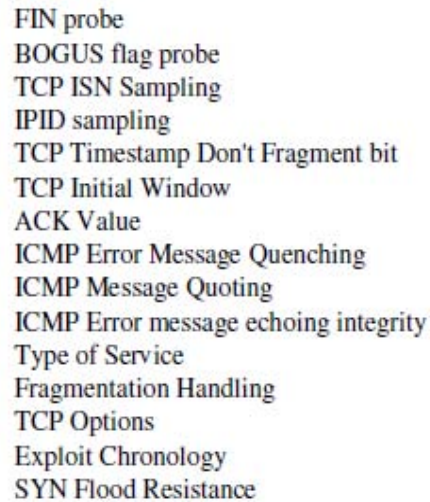
from various fingerprinting scans [Fyo02]. Nmap sends TCP, UDP, and ICMP scans to open and closed ports. These scans exploit the ambiguities in the standard protocol RFCs [Fyo02]. Nmap listens for the response and compares those responses with the profiles stored in its database.

The Nmap program uses a wide range of TCP/IP stack fingerprinting methods. For sequence generation scans, Nmap sends TCP SYN packets to the same port on a host but varies the window size, timestamp, MSS field, and several others to solicit different responses to each scan in order to more accurately determine the specifications of the implemented TCP/IP stack. A 'TCP ping' is a TCP packet with the ACK flag set with destination port of 80 [Wol02]. TCP pings can sometimes bypass stateful firewall rule sets [Wol02] and force a web server to decide how to handle a session acknowledgement when the server did not initiate the session. Some operating systems drop the ACK response while others send back a RST message [Fyo02].

Timing-related scans in Nmap can differentiate between TCP Tahoe or TCP Reno TCP/IP stacks [WSM04]. Almost all of the TCP options can be set and changed using Nmap as well as changing the port number to target with the specific TCP scan [Wol02].

ICMP scanning also plays an important role in Nmap OS detection. An IE (ICMP Echo) test by Nmap sends two ICMP echo request packets to the target host. The first ICMP message has the IP DF bit set, the TOS bit set to zero, a code of nine – which is normally set to zero, the sequence number as 295, and a random IP ID and payload [Fyo02]. Nmap also sends varying ICMP Requests messages such as the ICMP Timestamp Request message to help identify the operating system [Wol02].

Figure 16 provides a quick list of some of fingerprinting techniques that Nmap

uses to fingerprint a host, including the operating system and services.

FIN probe
BOGUS flag probe
TCP ISN Sampling
IPID sampling
TCP Timestamp Don't Fragment bit
TCP Initial Window
ACK Value
ICMP Error Message Quenching
ICMP Message Quoting
ICMP Error message echoing integrity
Type of Service
Fragmentation Handling
TCP Options
Exploit Chronology
SYN Flood Resistance

Figure 16: Common Nmap Fingerprinting Techniques[Kol05]

Nmap uses a SYN scan to detect open ports on the target. Nmap takes the list of

now known open ports on the target and connects to them by using a protocol list referred

to as a Service scan. The protocol list allows Nmap to detect the running application

which can disclose the underlying target's operating system.

Nmap produces two separate tests – the OS Class (direct operating system

identification) and Service (implied or direct operating system identification)

fingerprinting features of Nmap. These two separate Nmap scans are referred to as the

Nmap OS Class test and the Nmap Service test. An aggressive OS fingerprinting scan

uses both tests as shown by the following command: nmap –p 1-65535 –T4 –O –A –v –

sV –max-os-tries 10 –PE –PS22,25,80 –PA21,23,80,3389 –fuzzy –osscan-guess

10.1.2.67-70. The parameters used in the Nmap command are detailed below:

- -p 1-65535: identifies the port range to scan
- -T4: enables more aggressive timing for quicker response
- -O: enables operating system fingerprinting

- -A: enables version detection
- -v: enables verbosity to list additional operating system related details
- -sV: enables server version fingerprinting
- --max-os-tries 10: increases the number of operating system queries to ten
- --fuzzy and –osscan-guess: forces Nmap to make are more aggressive guess

The parameters listed above, enable aggressive TCP/IP and Service scanning.

The max-os-tries parameter increases the number of operating system detection queries

Nmap makes from the default two to ten (to allow for better correlation between

fingerprints and known hosts in Nmap's database). Both the –fuzzy and –osscan-guess

parameters forces Nmap to show the closest match in its signature base for the operating

system detected. The –fuzzy and –osscan-guess parameters increases the effectiveness of

Nmap's operating system detection as is evidenced by the results in Section 5.2.1.

2.4.2   Nessus

Nessus is another powerful tool that does multiple OS detection schemes. Nessus

uses over fourteen advanced operating system fingerprinting techniques [Gul09]. These

fingerprinting techniques include: banner grabbing, HTTP requests, HTTP responses,

SinFP algorithm (a new algorithm that detects the operating system using a single open

port response), SMB, ICMP (using Xprobe techniques), and several others.

Nessus version 4.2.2 is installed on January 14, 2011 and for use during the

obfuscation effectiveness experiment. Nessus is selected because Nessus incorporates

several fingerprinting agent techniques. Along with doing several TCP/IP scans similar

to Nmap's TCP/IP scans, Nessus uses the following techniques to detect the operating

system and services [Gul09]:

- os_fingerprint_html: uses HTML (Hyper-Text Markup Language) content returned by certain HTTP requests to fingerprint the remote operating system.

32

- os_fingerprint_http: uses the remote web server signature to infer the version of Windows or Linux running on the target.
- os_fingerprint_sinfp: uses the SinFP TCP/IP fingerprinting algorithm and only requires one open port to fingerprint an operating system.
- os_fingerprint_smb: identifies the operating system based on Windows SMB queries.
- os_fingerprint_ssh: identifies the remote OS by the SSH banner.
- os_fingerprint_xprobe: identifies the operating system type and version by sending incorrect ICMP requests.

The os_fingerprint_html script allows Nessus to use HTML content returned by

HTTP Response packets to identify the operating system which is very useful against

Microsoft IIS web servers that have default home pages or error pages. The

os_fingerprint_http script uses the server field in an HTTP Response packet as well as

other distinguishing HTTP header fields to identify the OS. The os_fingerprint_smb

script identifies the OS based on the differences between parameters used for a Linux and

Windows computer as well as looking for Native OS fields that are returned from certain

SMB queries.

Nessus emulates Xprobe, another popular operating system fingerprinting

program so Xprobe was eliminated as a tool needed to test against. The Nessus scripts

indicate how Nessus attempts to correlate running services on a target to the underlying

operating system running on that target. The SinFP algorithm is one of the newer

techniques used to detect a target's operating system that Nessus incorporates into its

operating system detection process.

The Nessus policy that is created to fingerprint each host enables all possible

scanning options, uses the "default" Port Scan Range, and uses all plugins available when

Nessus was downloaded. This policy allows for a comprehensive scan against the target

host. Nessus, similar to Nmap, returns two separate test results – Nessus OS Class test

and Nessus Service test – and each test is used to identify the target's operating system.

2.4.3   Xprobe

Another robust and popular fingerprinting tool is called Xprobe. Nessus

implements the Xprobe technique with its os_fingerprint_xprobe.nals script [Gul09].

Ofir Arkin developed Xprobe after extensively studying the ICMP protocol and how

ICMP can be utilized to determine host-level information [Ark02] [Ark01]. Xprobe is

capable of determining the host's operating system by only sending one datagram

[Ark02]. Xprobe differs from Nmap because Nmap relies heavily on TCP while Xprobe

relies solely on ICMP.

Xprobe, unlike Nmap, does not use malformed packets but instead relies

completely on the response from standard ICMP Request packets [Ark02]. One of the

more powerful scans Xprobe performs deals with ICMP error messages. An operating

system's implementation of the TCP/IP stack varies dramatically when responding to

ICMP error messages. An ICMP error message is made up of the IP header and at least

the first eight bytes of the IP packet payload [Ark02]. Some operating system echo back

more than just the first eight bytes of the IP packet payload. The ICMP specification in

RFC 1122 does not dictate the size of the echoed payload [Ark02]; therefore each

operating system vendor has their own unique implementation.

As is the case of a non-zero value in the code field for an ICMP ping packet, some

operating systems ignore certain RFC specifications for ICMP traffic. The specification

in the RFC states that the code value provided in the request should be echoed back in the

reply. Both Windows and Novell operating systems instead echo back a zero for the code value [Kol05]. Xprobe uses several IP header fields with different types of ICMP messages to identify the host.

2.4.4   Languard

The Languard Network Security Scanner (LNSS) tool is designed for network administrators to quickly identify hosts on the LAN. The LNSS tool does banner grabbing, SNMP scans, SMB scans, and Null session scans [Kol05]. Some of these scans are also possible from both Xprobe, Nessus, and Nmap, while the Null session scan is becoming more obsolete as newer operating systems block these requests.

2.4.5   p0f

A popular passive fingerprinting tool is called p0f. p0f relies on three different types of TCP packets to determine the operating system of the host – SYN, SYN+ACK, and RST+ [Kol05]. Each type of test utilizes a larger range of associated fingerprint files to identify the host OS. The same fingerprinting techniques used in p0f have led to actual network attack tools such as Ettercap and Siphon [Kol05]. Since p0f is a passive fingerprinting tool, p0f waits to receive TCP traffic from the host without initiating the communication. The techniques employed by p0f are used by Nmap and Nessus during their active fingerprint scanning.

2.5   Obfuscations Approaches

Two separate approaches exist to obfuscate host-level information. Network-based host obfuscation obfuscates or normalizes host information at the intra-network

level by placing an obfuscation device that intercepts and obfuscates all network traffic leaving the local area network. A host-based approach concentrates on obfuscating hosts using a one-on-one approach instead of obfuscating all the hosts' packets from an entire subnet or local area network and is not limited to obfuscating only packets that leave the local area network.

2.5.1 Network-Based Obfuscation Techniques

Host obfuscation provides a proactive approach to hardening the TCP/IP stack of a host [KaS10]. Most of the network-based host-level obfuscation techniques fall into one of three categories: a transport scrubber, a proxy server, or a traffic normalizer.

Transport scrubbers attempt to remove protocol related ambiguities from network traffic. The research done with transport scrubbing implements a new network-level device placed on the inside interface of a firewall or IDS [SMJ00].   By placing the transport scrubber as a bridged device between all the hosts on the LAN and the firewall or IDS, the transport scrubber can remove all the LAN-generated protocol ambiguities allowing the IDS to see common, standard protocol implementations [SMJ00].

To date, no further research has been found that has combined the transport scrubber device with a firewall or IDS. This research would be crucial in providing a more efficient and effective host-level obfuscation. The transport scrubbers, or fingerprint scrubbers, become another bottleneck in network traffic and another possible attack vector. Current transport scrubbers run on the FreeBSD version of Linux and only operate on the TCP and IP layers [SMJ00]. Transport scrubbing has been found to be effective against several fingerprinting techniques [SMJ00] but has not been fully tested

against a fingerprinting program that utilizes multiple fingerprinting techniques to identify the host's operating system, such as Nmap and Nessus.

Traffic normalization is a similar approach to transport scrubbing. The goal of traffic normalization is to preserve well-behaved network protocols while cleaning any misbehaving traffic [HPK04]. Both traffic normalization and transport scrubbing change field values in IP headers, specifically: fragment offset, DF flag, and TTL [HPK04]. The difference lies in that transport scrubbers consist of two interfaces – one interface is considered trusted and the other untrusted while traffic normalization does not make a distinction [HPK04].

Both traffic normalization and transport scrubbing can defeat stealthy port scanning techniques because normalization and scrubbing remove ambiguities from all packets and does not filter packets. Traffic normalization faces the same inherit benefits and disadvantages as transport scrubbing. Traffic normalizers can be bypassed in the case of a cold restart of the traffic normalizer. A session that is initiated prior to the startup of the traffic normalizer will continue without being normalized (to not interfere with TCP and IP checksums for that session's traffic) [HPK04]. Normalizers struggle to accurately determine when a TCP connection has been reset [HPK04]. Traffic normalizers and scrubbers remove a limited set of protocol ambiguities.

A proxy server, used for host obfuscation, operates by making a request to the host and responding back to the original requester with a modified response from the host. If the proxy server receives a packet (e.g., a fingerprinting scan), the proxy server does not forward the packet onto the target host, but instead initiates a new connection directly to the host requesting the information. When the host responds back to the proxy

server, the proxy server then eliminates a pre-configured set of host information and sends the modified reply back to the original requester. The proxy server method decreases network performance by introducing a new bottleneck of network traffic and only removes a limited set of ambiguities [MaB10].

2.5.1.1   Network-Based Obfuscation Devices

The primary focus of host obfuscation has been dedicated at the network level for Windows hosts. These network level defenses include: Intrusion Detection Systems (IDS), Intrusion Protection Systems (IPS), firewalls, and proxies utilizing the obfuscation methods previously detailed. Host-level information cannot be protected by relying on network level protection alone because network-level protection is not efficient or robust at protecting multiple host operating systems.

Network devices may address a few of the external attack vectors of external fingerprinting techniques but can be bypassed. Several fingerprinting programs identify firewall rule sets that then allow the attacker to craft special fingerprinting packets to bypass the firewall rule sets [Wol02]. A fingerprinting technique, used to bypass a firewall or IDS, uses slow and distributed scans, which provide a stealthy way to fingerprint a host behind a firewall [Yui06].

Network security devices are dependent on their TCP/IP stack implementation which differs from the possible wide range of host operating system implementations located on the local area network behind the network-level device [HPK04]. Since the protocol standards do not accurately specify the behavior of the protocols, the network-level device implements different aspects of the undefined protocols specifications.

A firewall or IDS does not know how to treat network traffic similar to how the traffic will be treated by the host operating system [Tal03]. The protection mechanisms implemented on a network-level security device vary greatly and are dependent on the configuration established by a network administrator [LiT08].

The benefits of obfuscating host information at the network level are primarily the ease of implementation, manageability, and utilizing existing network security resources. Network-level security devices are already examining all the traffic entering and leaving a LAN. Obfuscating at the network-level can better ensure that all hosts inside the LAN have at least some type of host-level obfuscation but cannot ensure that all network traffic from each host is obfuscated.

2.5.2 Host-Based Obfuscation Techniques

Two separate host-based implementations are examined – obfuscating the TCP/IP stack implementation of the host and protocol stack virtualization. Current host-based obfuscation programs deal with changing aspects of the TCP/IP stack implementation and primarily not with protocol stack virtualization. Changes to the TCP/IP protocol stack are dependent upon the operating system and the specific program used. Protocol stack virtualization is, to date, more of a proof of concept and not actually a fully functional implementation for host obfuscation [LiT08].

The goal of protocol stack virtualization is to create separate instances of the protocol stack for each service [LiT08]. Each service uses a distinct set of identifiers so that the host's network interface card (NIC) can send the packet to the correct service. Protocol stack virtualization's main advantage is obfuscating which services actually

belong to a specific host to provide the illusion of many additional hosts, each running a single service [LiT08]. This approach helps uncorrelate certain fingerprinting scans that are typically used to validate each other. Some fingerprinting programs, such as Xprobe and SinFP, pride themselves on the ability to accurately determine the operating system of the host using minimal packets. Reducing the number of useful open port responses might not provide any obfuscation benefits beyond deceiving an attacker regarding the services found on each host and the number of hosts on the LAN. Instead, the research behind protocol stack virtualization encourages combining protocol virtualization with either TCP scrubbing or traffic normalization [LiT08].

2.5.2.1   *Host-Based Obfuscation Tools*

Host-based obfuscation programs are located directly on the host device and intercept the network traffic before the traffic leaves the host. Host-based approaches, similar to scrubbing and normalization, attempt to remove ambiguities and identifying features found in protocol implementations.

Most host-based obfuscation programs run only on a Linux operating system. A well-documented host-based obfuscation program is called IP Personality. IP Personality is effective at fooling some Nmap scans by changing the characteristics of the TCP/IP stack implementation of the host [RoS01]. IP Personality can change the TCP Initial Sequence Number (ISN), the TCP initial window size, the TCP options (types, values and their order), the IP ID numbers, and can answer some pathological TCP and UDP packets [RoS01].

IP Personality allows a user to select which operating system the host should impersonate by automatically changing the necessary TCP/IP stack implementation. IP Personality hooks into the Linux kernel using the netfilter hook. IP Personality also works as a network-based host obfuscator by installing IP Personality on a Linux firewall or IDS system [Ber03]. In this regard, IP Personality can be directly installed on all Linux hosts as well as installed on a Linux firewall which will then obfuscate the rest of the network traffic originating from non-Linux operating systems including Windows operating systems. This dual approach helps cover external fingerprinting vectors but does not address malware or internal vectors towards non-Linux machines. IP Personality is effective against standard TCP fingerprinting scans but does not provide the same obfuscation benefits against ICMP-related scans or Application layer protocols. IP Personality is primarily an IP scrubber [KaS10] because of its focus on scrubbing or normalizing the IP header.

Stealth patch is another Linux kernel TCP anti-fingerprinting tool. This tool does not obfuscate the TCP/IP stack implementation but merely discards the following TCP/IP packets: packets with both the SYN and FIN flag activated, if the packet has the reserved bit set in the TCP header or none of the TCP flags set, and packets with the FIN, PSH and URG flags set [Ber03].

Additional Linux-based obfuscation tools such as IPLog and Blackhole can implement TCP/IP stack changes similar to IP Personality by changing values of the TCP/IP stack and dropping malformed packets instead of allowing the operating system to reply. These additional tools are still useful, but provide more limited obfuscation

techniques and diversions [Ber03]. Several fingerprinting techniques do not rely on malformed packets to obtain host-level information.

The tools listed primarily obfuscate only portions of the TCP/IP stack and only run on Linux operating systems. In order to obfuscate banner grabbing and HTTP fingerprinting techniques, manual changes must be made to each specific application. In the case of banner grabbing, the application that is providing the banner must be manually changed which might not be a realistic option. To defeat HTTP fingerprinting four main items must be taken into account: changing the HTTP Server field, re-arranging HTTP headers, customizing HTTP error codes, and possibly even using an HTTP server plug-in [Sha04].

The only Windows-based obfuscation program identified is called OSfuscate. OSfuscate changes seven Windows registry settings as shown in Figure 17. These settings change how the Windows operating system implements portions of the TCP/IP stack. OSfuscate attempts to impersonate a different operating system similar to IP Personalilty. OSfuscate changes some of the more heavily utilized TCP/IP header fields for fingerprinting, as shown in Figure 17, including: TTL, MTU and the window size. OSfuscate focuses on TCP and IP protocol fingerprinting methods and does not obfuscate additional protocols – such as ICMP, UDP, HTTP, and SMB.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\TcpUseRFC1122UrgentPointer
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\TcpWindowSize
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\SackOpts
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\*\MTU
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\DefaultTTL
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Tcp1323Opts
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\EnablePMTUDiscovery
```
Figure 17: OSfuscate Registry Changes [Cre08]

2.5.3 Polymorphic Approach to Host-Level Obfuscation

Several research projects and efforts have been directed towards a polymorphic approach to obfuscating a host's information. A well documented polymorphic project is the DyNAT project [MPS02]. Polymorphic approaches continually change the 'location' of a particular host by changing the host's IP address, MAC address, and/or port numbers used for certain applications.

This approach requires rigid time synchronization in order for a "trusted" host to be able to connect to another "trusted" host. Each host needs to know the timing and mechanism used to change the network configuration of each other host. This approach protects a host's information by invalidating the location of the host, making the host unreachable from another un-trusted source.

These approaches do not actually obfuscate the host operating system or service (although some, like port hopping [LeT04] do obfuscate the current port number a service is utilizing). The inherit problems with these architectures is that all the "trusted" hosts are considered trusted and know the polymorphic mechanism in place. If a host is compromised, then the polymorphic mechanism becomes compromised in respect to all hosts which use that the same polymorphic mechanism. Two of the fingerprinting vectors previously outlined – malware and internal – are not addressed by polymorphic approaches. Likewise some of the polymorphic research projects can dramatically increase network traffic – such as, increased Address Resolution Protocol (ARP) traffic when the MAC address changes. DNS mapping considerations have also yet to be fully considered.

43

2.5.4   Comparison of Current Obfuscation Programs

Current obfuscation programs and tools are limited in their scope. Table 1 shows

the type of network traffic that each obfuscation program is able to intercept and

obfuscate. IP Personality and OSfuscate are the only two programs capable of

obfuscating packets originating from a Windows operating system. This research focuses

on host-based obfuscation programs to ensure that all network packets would be

obfuscated by obfuscating each packet as it leaves and enters a host. OSfuscate is

selected to compare against the results of the SNOS program developed during this

research because OSfuscate is the only program that runs on a Windows operating system

and claims to obfuscate the Windows operating system.

Table 1: Operating System Current Obfuscation Programs Can Obfuscate

	Network-Based	Host-Based
IP Personality	Linux / Windows	Linux
Stealth Patch	N/A	Linux
IPLog	N/A	Linux
Blackhole	N/A	Linux
OSfuscate	N/A	Windows

Table 2 shows which protocols each current obfuscation program can obfuscate.

Current programs only modify the IP, TCP, UDP, and ICMP protocols. Table 2 shows

the limitations of each program, particularly the limitations of OSfuscate. OSfuscate,

Stealth Patch, IPLog, and Blackhole only obfuscate the TCP and IP protocols. None of

them obfuscate Application-layer protocols.

Table 2: Protocols Current Obfuscation Programs Can Obfuscate

	IP	TCP	UDP	ICMP	HTTP	SMTP	SMB	SSH	DNS
IP Personality	Y	Y	Y	Y	N	N	N	N	N
Stealth Patch	Y	Y	N	N	N	N	N	N	N
IPLog	Y	Y	N	N	N	N	N	N	N
Blackhole	Y	Y	N	N	N	N	N	N	N
OSfuscate	Y	Y	N	N	N	N	N	N	N

2.6 Achieving Host Operating System Obfuscation

Effective host obfuscation must follow the basic deception pattern. Poor planning or implementation can leave the host more vulnerable than before the obfuscation technique was applied [Bec01]. Obfuscation techniques that deal with the TCP portion of a packet must accurately safeguard TCP Sequence Numbers, checksums, and congestion window size [PaF01]. The initial sequence number must still be a random generated number; otherwise, TCP hijacking can occur by predicting the sequence number. Network degradation also presents a possible hurdle into developing an effective host-based obfuscation technique. Host-based obfuscation requires an additional program to modify the network packets leaving the host and requires additional processing time for each created packet before the packet is sent onto the network. Disadvantages in implementing host-based obfuscation are that each host must be configured and managed independently.

Obfuscation techniques affect a host's resources even though no fingerprinting activity is present and might never be present [WJS07]. Depending upon the obfuscation method, the host's resources might become more efficiently utilized during a large

45

fingerprinting scan attempt if the obfuscation technique merely drops all malformed packets upon arrival [RoS01].

Planning a sophisticated TCP/IP stack implementation, while not addressing banner messages, can nullify the obfuscation technique and must be taken into affect as well. Obfuscation that does not limit the amount of chatter from a host can only be minimally effective – some of the sensitive host-level information is sent as a broadcast message to all hosts on the same subnet [Kol05]. Reducing broadcast messages can particularly help limit the effectiveness of passive fingerprinting. The obfuscation must be verifiable from all protocols within the network traffic from a host.

Host-based obfuscation provides another layer of information protection in addition to network-based protection. To date, most research focuses on network-based obfuscation or the Linux operating system family for host-based obfuscation. Host-based obfuscation of the Windows operating system family has received minimal attention. By obfuscating all network packets, and all the protocols within a network packet, from a Windows operating system, this research attempts to provide a system wide obfuscation approach to protecting the identity of the Windows operating system and services.

III. Methodology

This chapter outlines the methodology used including the goals, problems, and questions addressed by the Host-Based Systemic Network Obfuscation System. The goals and approach are discussed in Section 3.1. Section 3.2 presents the system boundaries. The approach and description of the components, parameters, and metrics are also discussed in this chapter, in addition to the experimental process utilized.

3.1   Problem Definition

Host devices often divulge sensitive information about themselves. The exposed sensitive host information leaves the device more vulnerable to attack vectors. Host devices divulge sensitive information directly –services publicly identify their type and version information – or indirectly – inferred by how a service or the underlying operating system creates or modifies a network packet. These two closely related problems make the host more susceptible to attacks and exploits.

Chapter 2 discussed the background of this problem and some of the work that had been done to correct this problem. Current obfuscation programs and tools only obfuscate a very limited set of protocols and only a few are capable of obfuscating packets created from a Windows operating system. The limited protocol set used by these programs result in the obfuscation process becoming useless because other un-obfuscated protocols still identify the host's operating system.

3.1.1 Goals

This research hypothesizes that a host-based obfuscation approach can be an effective tool to defeat fingerprinting programs by obfuscating a host's entire network packet. The Systemic Network Obfuscation System (SNOS) is developed to defeat multiple fingerprinting techniques and provide a constant deceptive response. The SNOS program obfuscates all OSI layer protocols within a network packet to provide a consistent and complete deception. The network performance degradation caused by running the SNOS program is also analyzed.

3.1.2 Experimental Setup

The experimental setup uses standard Windows host services including: email, web, and SharePoint. The Windows operating system provides the network services and packet creation which is evaluated to determine the effectiveness and performance of the SNOS program, a benchmark trial, and OSfuscate.

All four host devices are virtual computers running on a VMware ESXi server during both experiments – obfuscation effectiveness and network latency. Virtualization allows the experiments to run with exactly the same configuration to eliminate variability in configuration within each host. Each experiment is repeated multiple times and started with the exact same configuration for each repetition. The only variable difference between each trial is whether an obfuscation program – OSfuscate or SNOS – is running or not.

Windows XP and Windows Server 2003 are the two Windows operating systems. According to a January 2011 study, Windows XP accounts for 45.3% of all operating systems used and Windows Server 2003 has the highest Windows server usage [W3s11].

As shown in Figure 18, the infrastructure topology consists of a workstation, an email server, a SharePoint Services server, and a web server. The workstation is running Windows XP SP3 and only offers SSH and SMB as a network service. Windows XP and Windows Server 2003 enable the SMB service by default. The remaining three host configurations use the Windows Server 2003 operating system. The email server runs Microsoft Exchange Server 2003. The SharePoint server runs Microsoft SharePoint Services 3.0 and the web server runs Apache for Windows 2.0.

User guides and default installation manuals are used to configure and install the software on all four hosts. The Windows operating system, XP and Server 2003, are installed with the default selections from the installation disks. The Exchange server is configured using Microsoft's Active Directory installation guide and the Exchange Server 2003 installation guide [Mic05]. SharePoint is installed on the SharePoint server using the Windows SharePoint Services 3.0 installation guide [Mic09]. Apache is installed on the web server using the Apache for Windows installation guide [ASF11].

The Performance Monitor (Cacti) computer monitors network performance generated from each trial by the four hosts and records the results of the network latency experiment. The Nessus/Nmap computer uses the fingerprinting programs Nmap and Nessus to fingerprint each host. The Tcpreplay computer generates the network traffic load discussed in Section 3.5.2 for each host. The Cacti and Tcpreplay computers

analyze and record the network latency experiment. The Nessus/Nmap computer is used during the obfuscation effectiveness experiment.
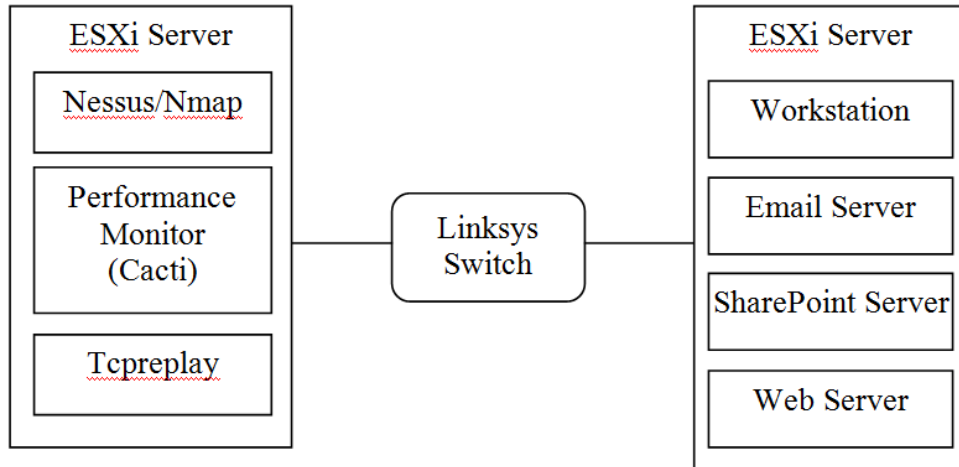
Figure 18: Physical Setup of Experiment

Figure 19 shows the experimental methodology used in this research. A test produces a unique result. The obfuscation effectiveness experiment uses four tests to compare against each trial. Each trial produces four test results for each of the four hosts. The network latency experiment used one test, roundtrip time. A trial is defined as a complete set of tests for all four hosts. The obfuscation effectiveness experiment has three trials – no obfuscation (benchmark), OSfuscate, and SNOS. Trial results are used to compare the effectiveness of each obfuscation program with the benchmark.

An experimental run, or repetition, is defined as a complete set of trials. The obfuscation effectiveness metric has forty runs and each run is composed of the three separate trials – benchmark, OSfuscate, and SNOS. An experiment constitutes a complete set of repetitions. Two independent metrics, obfuscation effectiveness and network latency, are identified; therefore, two separate experiments are run for each metric. Figure 19 shows the three trials of the obfuscation effectiveness experiment that

are repeated forty times to complete the experiment. OSfuscate is not tested during the network latency experiment because OSfuscate only modifies Windows registry settings and is not a live program intercepting and modifying packets to affect network latency.
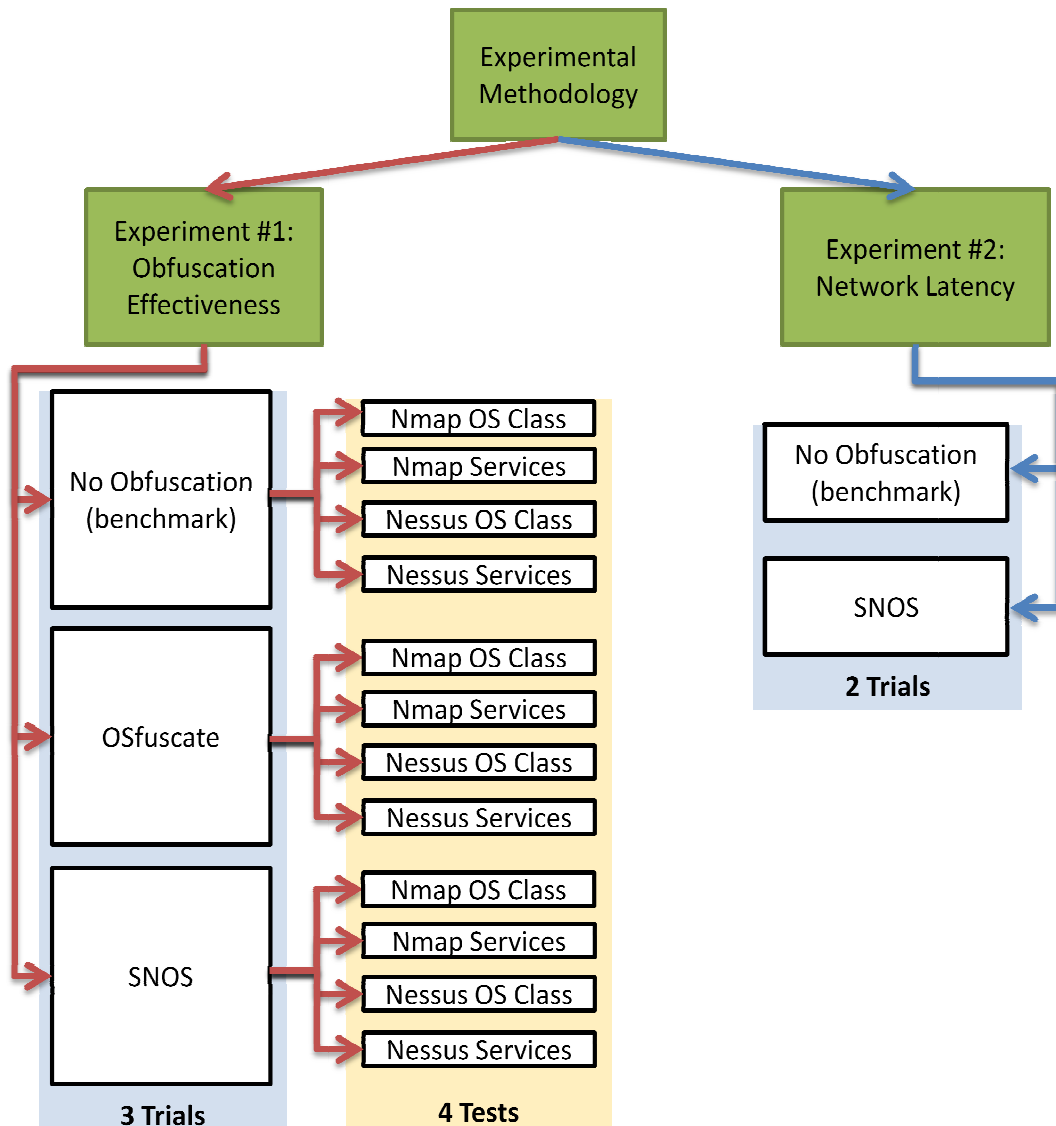
Figure 19: Experiment Methodology

3.2   System Boundaries

The system under test includes the complete set of components – both software

and hardware – involved in providing network traffic obfuscation. The system under test

is a host-based network traffic obfuscation system, SNOS. Figure 20 shows an overview

of the system. The Systemic Network Obfuscation System is a white box system –

meaning that the components of the system could be examined. The system under test

consists of several components – the host, network interface card, an adversary, and

obfuscation tool. The obfuscation program is identified as a component under test. The

NIC, host applications – including the operating system and services – and virtualization

are considered black box objects because only the inputs and outputs of these

components are used.
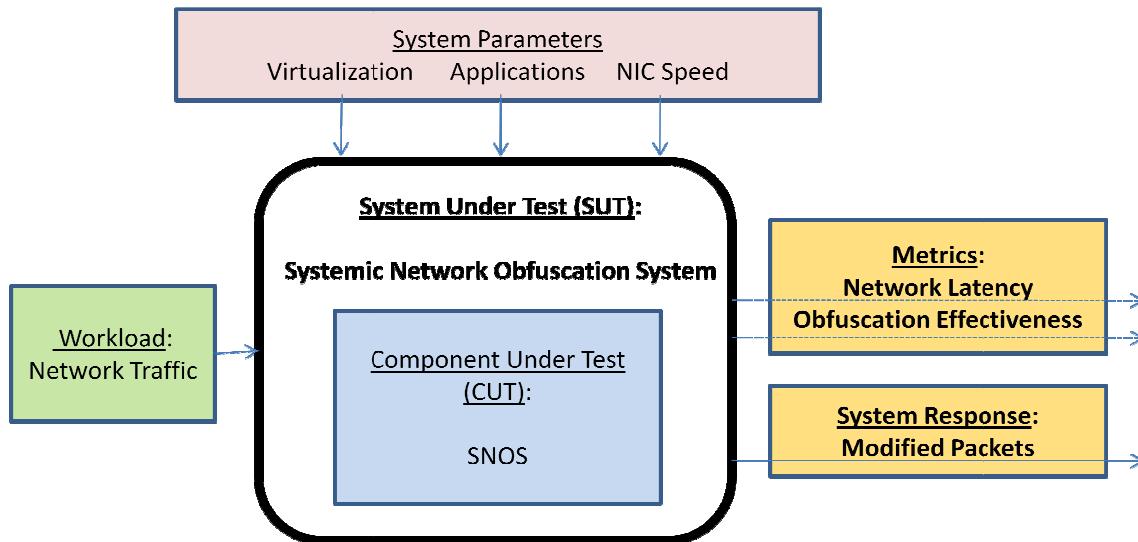
Figure 20: Systemic Network Obfuscation System

3.3   System Services

The primary service of the Systemic Network Obfuscation System (SNOS)

provides network packet obfuscation. The outcomes of this service are a success or

failure – either the host cannot be fingerprinted or the hosts are still able to be fingerprinted. A success is measured only when the fingerprinting program, Nmap or Nessus, is not able to identify the host operating system; otherwise, the obfuscation service fails.

3.3.1 Service By Component

The component under test, SNOS, provides two services – header and payload obfuscation, and TCP session state preservation. SNOS uses a Network Driver Interface Specification (NDIS) Intermediate driver called Winpkfilter. The NDIS Intermediate driver functions on the Logical Link Control (LLC) layer of the Open Systems Interconnection (OSI) model which corresponds to the layer between the protocol and the miniport layers of a Windows operating system shown in Figure 21.

Windows operating systems, since Windows XP and Server 2003, implement a section within the Network Device Implementation Standard (NDIS) stack called an Intermediate driver. NDIS Intermediate drivers can intercept all network traffic in either direction – incoming or outgoing – and has access to the entire TCP/IP model, except the physical layer, as implemented on a Windows Operating System. A network packet being delivered to the host works up through the layers in Figure 21 from the NIC driver, miniport layer, to the NDIS Intermediate layer and then to the protocol layer. A packet sent by the host works down the layers.

Winpkfilter intercepts all network traffic sent from the host as the traffic passes through the NDIS Intermediate layer. SNOS interfaces with the Winpkfilter driver to modify the packets before sending them out on the wire or passing them up to the

protocol layer. The Winpkfilter driver comes with a set of Application Programming Interfaces (API) that allow SNOS to integrate with the NDIS Intermediate driver. SNOS constantly is pulling the packets from the Winpkfilter queue.
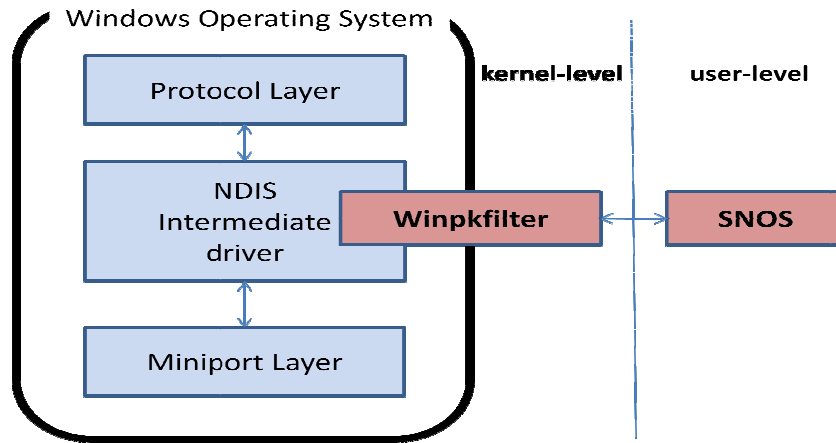
Figure 21: SNOS Implementation into Windows

The Winpkfilter driver runs as a kernel-level driver, and SNOS runs as a user-level program as shown Figure 21. SNOS uses the NDIS Intermediate driver layer, through the Winpkfilter driver – to modify packets for each individual host running a Windows operating system. By running through a NDIS Intermediate driver, SNOS is capable of modifying layers two through seven of the OSI model. By installing SNOS on each individual host, SNOS modifies each host's network traffic individually instead of operating on the subnet or local area network level.

3.4   Performance Metrics

Performance metrics measure the effectiveness of the system. Network latency measures the performance by measuring the amount of time a packet takes to reach the target and have the host create a response back to the performance monitoring (Cacti)

computer. The latency measured is the roundtrip time (RTT) recorded by the Cacti performance monitoring computer.

Cacti is a free Linux-based tool used to monitor network performance. A Perl script sends out ten ICMP echo request packets to each host and calculate the mean roundtrip time until the ICMP echo reply packets arrive. The Perl script is continuously run to collect roundtrip times throughout an entire experimental trial repetition. The Perl script calculates the mean for the ten ICMP packets which is used as the recorded result.

The first trial of the network latency experiment is done without SNOS and is used as a benchmark to compare any difference in latency experienced when SNOS is enabled. The RTT is measured in the number of microseconds for the response packet to reach the Cacti computer from the host.

The obfuscation effectiveness metric measures the obfuscation program's ability to correctly obfuscate packets so that fingerprinting programs are not able to identify the host's operating system. The effectiveness of SNOS is measured against two independent trials – one with no obfuscation (the benchmark), and the other with OSfuscate. OSfuscate is a Windows program that claims to obfuscate the underlying Windows Operating System as discussed in Section 2.5.2.1.

The first trial in the obfuscation effectiveness experiment fingerprints the host without any obfuscation program and provides a benchmark to compare against the second and third trials when either OSfuscate or SNOS is enabled. The obfuscation effectiveness measurement is recorded in terms of the successful number of times a fingerprinting method is defeated out of the total number of fingerprinting tries attempted.

3.5   Parameters

The parameters of a system directly affect the performance of the system. Parameters are divided into two sets – system parameters and workload parameters. Parameters are aspects of a system that can cause variation and change the results of an experiment.

3.5.1   System Parameters

The Systemic Network Obfuscation System has several system parameters. A virtualized host has degraded network throughput and latency caused by timing delays associated with virtualization overhead.   The rationale for using virtualization in this system is detailed in Section 3.1.2. Since all the trials within each experiment use virtualization, virtualization does not affect the results between trials.

The applications, or services, running on the host device also affect performance. These applications initiate the network packet creation process and include application layer headers that SNOS modifies to counter operating system fingerprinting techniques. Each application creates its own application layer header and payload. The applications, or system parameters, include a Microsoft SharePoint server, an email server, a web server, and a workstation running an SSH (Secure Shell) server.

Microsoft Exchange is the selected email server platform. According to recent research done by Ferris Research, Microsoft Exchange had 66% of the email market share worldwide in 2008 [Fer08].   Apache is used as the web server application. Netcraft reported that as of January 2009, the Apache web server held 50% of the worldwide market share while Microsoft Internet Information Services (IIS) held 32%

[Net09]. Microsoft IIS is an essential part of Microsoft SharePoint Services and is tested as a part of the SharePoint server host. SharePoint Services is selected because 83% of corporations use or plan to use SharePoint and almost half of those corporations utilize SharePoint primarily as a file sharing server [Fra09].

Additional parameters include NIC speed, CPU speed, and memory size. These additional parameters are control variables and do not change throughout the experiment. The applications running on a particular host also function as control variables because each host configuration remained constant throughout both experiments.

3.5.2 Workload Parameters

Nmap and Nessus fingerprinting programs are used to fingerprint each host. The same commands and parameters – detailed in Section 4.2.1 and 4.2.2 – are used for Nmap and Nessus during the fingerprinting process. The network traffic generated by each program throughout the obfuscation effectiveness experiment remains constant and is considered a control variable.

The Tcpreplay host in Figure 18 sent out specially-crafted packets to each host during the network latency experiment. The Tcpreplay pcap files were originally captured during a simulated real world experience – the Cyber Defense Exercise (CDX) from April 2010 at the Air Force Institute of Technology. The original pcap file was captured from the day with the heaviest network load during the CDX. Using Tcpwrite and Wireshark, the pcap files are modified for each of the four hosts – Exchange, SharePoint, web, and workstation. The IP addresses, MAC addresses, and applicable services are changed to correctly match the target host device. For example, the custom

57

pcap file for the Exchange email server modified all SMTP (Simple Mail Transfer Protocol) traffic to be addressed to the Exchange email server host. Each host has its own specially crafted pcap file that is used by Tcpreplay to send to the host from the Tcpreplay computer.

The custom pcap file workloads are a synthetic workload because the original pcap file was captured from a specific working scenario and cannot represent all possible workload scenarios. A synthetic workload is the only feasible because each organization has a different infrastructure topology and configuration.

The Tcpreplay program sends the exact same packets to a host every time and is used to eliminate network traffic workload variance as a possible reason for individual results. Tcpreplay is constant throughout each trial and repetition for the network latency experiment.

3.6   Factors / Levels

A factor is a subset of parameters that are varied during the experiment. SNOS uses two factors – obfuscation and host applications. The obfuscation factor included three levels – no obfuscation, OSfuscate, and SNOS. The host application factor has four levels – email server, SharePoint server, web server and workstation.

Table 3 shows the twelve combinations of levels and factors for the obfuscation effectiveness metric. Each level, or trial, consists of four individual tests – Nmap OS Class, Nmap Services, Nessus OS Class, and Nessus Services – throughout the obfuscation effectiveness experiment. Forty-eight individual results, or sample points,

(twelve factor/level combinations multiplied by four tests) are achieved each time the experiment is repeated (experimental run).

Table 3: Obfuscation Effectiveness Experimental Factors and Levels

	No Obfuscation	OSfuscate (OSf)	SNOS
Email Server	No obfuscation / Email Server	OSf / Email Server	SNOS/ Email Server
SharePoint server	No obfuscation / SharePoint server	OSf / SharePoint Server	SNOS / SharePoint server
Web Server	No obfuscation / Web Server	OSf / Web Server	SNOS / Web Server
Workstation	No obfuscation / Workstation	OSf / Workstation	SNOS / Workstation

Table 4 shows the various combinations of levels and factors for the network latency metric. Each trial in the network latency experiment only has a single test, roundtrip time which produces eight individual results (eight factor/level combinations multiplied by one test) each time the experiment is repeated.

Table 4: Network Latency Experimental Factors and Levels

	No Obfuscation	SNOS
Email Server	No obfuscation / Email Server	SNOS/ Email Server
SharePoint Services server	No obfuscation / SharePoint Services server	SNOS / SharePoint Services server
Web Server	No obfuscation / Web Server	SNOS / Web Server
Workstation	No obfuscation / Workstation	SNOS / Workstation

Repeating the full experimental process several times reduces the variance of the results within each trial. A full factorial design using the factors above tests every combination of levels for each factor. Sufficient repetitions are performed when the variance between the results becomes consistent within each trial.

3.7   Evaluation Technique

The evaluation technique is emulation. Emulation is a simulation and a measurement of a real system.

3.8   Summary

The SNOS program is created to defeat fingerprinting techniques by obfuscating all the OSI layers of a network packet. The Systemic Network Obfuscation System uses a NDIS Intermediate driver to intercept and obfuscate the network packets. The component under test is the network traffic obfuscation program – SNOS. The parameters used to test the system include the network load and host applications. A full factorial design determines the effect of each factor during the obfuscation process of the SNOS program. The experiment is repeated to reduce variability and errors. The results and responses provide the necessary data to determine the effectiveness of the obfuscation program in defeating the fingerprinting programs.

IV. SNOS Program Design

This section provides detail regarding the design of the SNOS program, including why and how SNOS determines which protocols to obfuscate and what fields and/or payloads need obfuscation.

4.1   Systemic Network Obfuscation System

Network Driver Interface Specification (NDIS) encompasses several layers within the Windows network stack implementation. NDIS drivers can be used in one of three locations: the Protocol layer, the Intermediate driver layer, or the Miniport layer. The Miniport layer is where a network card's driver is located and the Protocol layer is the finished Protocol stack – the TCP/IP stack for this research – ready to send out through the network card. The NDIS Intermediate driver layer allows a program to intercept a completed network packet before it is sent on the wire or received by the destined application on the host.

Winpkfilter is a NDIS Intermediate driver that can intercept all network packets entering or leaving a host. Winpkfilter uses a queue to store packets so that any program using Winpkfilter can continuously grab the packets from the queue. The SNOS program loops through the Winpkfilter queue to grab all the intercepted packets. SNOS waits until a notification event is set within the Winpkfilter structure which indicates a packet is in the queue. SNOS then grabs the packet structure from Winpkfilter and directly modifies the structure by modifying the individual bits within it.

The Systemic Network Obfuscation System uses the Winpkfilter NDIS

Intermediate driver so all layers, except the physical layer, of the OSI model, shown in

Figure 22, could be modified as needed. The software SNOS needs to function includes:

the Winpkfilter driver and the .Net Framework 3.5 or greater. Winpkfilter and the .Net

Framework are installed on each host as part of the host's base virtual image so all of the

trials, including the benchmark, have these two programs installed and configured. The

only difference between the baseline and the SNOS trials is whether SNOS is running or

not.   An important step in building SNOS is determining which protocols need to be

obfuscated.

4.1.1   Protocols

A seemingly infinite list of protocols exists for various forms of network

communication. Protocols are used in layers two through 7 inclusive, as shown in Figure

22, of the OSI model. The physical layer represents the physical median that passes the

network traffic, such as a wireless signal or Ethernet cable and does not need obfuscation.

The colored layers in Figure 22 show the OSI layers used to obfuscate the network

traffic. The Data Link layer in SNOS ensures that the network traffic is using the Ethernet

protocol.

Figure 22: OSI Model

Protocols are selected using two criteria: 1) commonly used protocols on a network and 2) protocols used to fingerprint an operating system. A study based on three large organizations found that the top five protocols on the network were: HTTP, SMB, SSL, SSH, and SMTP [DFM06], as shown in Table 5. The five protocols identified are all Application Layer protocols and therefore the protocols from the other lower layers of the OSI model are also obfuscated. All five of these protocols use the IP Network Layer protocol and the TCP Transport Layer protocol.

Table 5: Protocol Usage Study

Port	Connection	% Connection
80 (http)	97,106,281	70.82%
445 (smb)	4,833,919	3.53%
443 (ssl)	3,206,369	2.344%
22 (ssh)	2,900,876	2.12%
25 (smtp)	2,533,617	1.85%

The protocols that fingerprinting programs use to identify a host's operating systems is the second criteria for selecting the protocols the SNOS program would obfuscate. The most common operating system fingerprinting techniques use three

protocols: IP, TCP, and ICMP [All07]. Additionally SNOS obfuscates protocols that

were identified in Chapter 2 and are used by the Nmap and Nessus fingerprinting

programs.

The SSL (Secure Sockets Layer) protocol did not need obfuscation (although the

lower layers in the OSI Model still did). SSH servers are primarily written to run on a

Linux operating system. OpenSSH, a Linux SSH server, can run on a Windows

operating system inside of the Cygwin environment. Cygwin is a Windows program

which allows native Linux programs to run on a Windows operating system by providing

Linux API calls. SNOS has an SSH obfuscation method but the method is not used

because the SSH server does not give away the underlying host operating system

throughout any of the tests.

4.1.2   SNOS Overview

Figure 23 shows an overview of how SNOS inspects the network packets to

determine if the packet needs to be obfuscated. Packets arrive at the NDIS Intermediate

driver layer, where SNOS uses Winpkfilter to intercept the packet before the packet

continues to the next NDIS layer in Figure 21. If the packet does not need obfuscation,

then the packet is sent to the next appropriate NDIS layer. For incoming packets, SNOS

uses Winpkfilter to send the packet up to the protocol layer in Figure 21, and for outgoing

packets, SNOS uses Winpkfilter to send the packet down to the miniport layer in Figure

21.
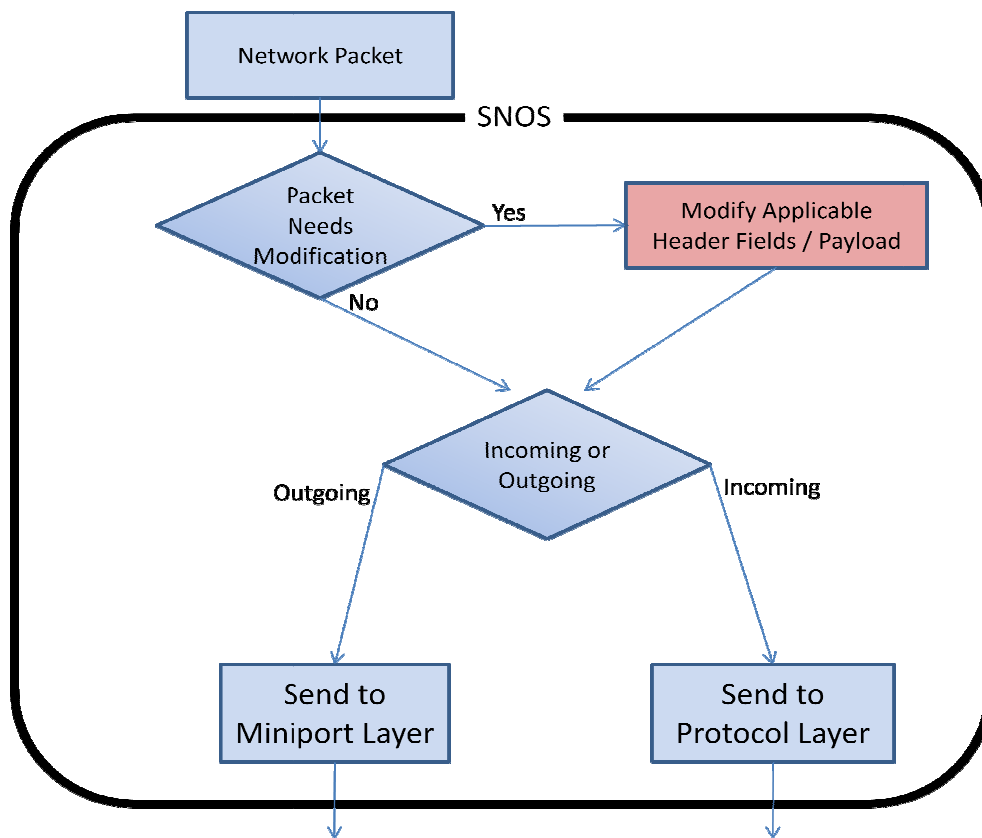
Figure 23: SNOS Decision Flow Overview

4.1.3   SNOS Packet Modification

SNOS packet modification commences when SNOS determines that the packet

needs obfuscation. Figure 24 shows a detailed view into the decision "Packet Needs

Modification" and "Modify Applicable Header Fields / Payload" objects from Figure 23.

To determine if the network packet needs obfuscation, SNOS first verifies that the packet

is an Ethernet packet and uses the IP protocol as the packets Network layer protocol.

SNOS focuses on IPv4 and does not consider IPv6 packets.

If the Network protocol is IP, then SNOS checks for Transport layer protocols

that need to be obfuscated. SNOS obfuscats protocol headers in the reverse order of

when SNOS checks for that protocol. For example, SNOS checks for the IP protocol

second in the obfuscation decision making process but does not actually obfuscate the IP header until the very end of the obfuscation process. This process ensures that checksum fields found in various protocols are accurate. The checksums are recalculated because the upper layer protocols and/or payload is modified making the old checksum obsolete. The new checksum is copied over the old checksum in the checksum header field of the protocol.

SNOS checks for ICMP packets at the same time Transport layer protocols are checked. The three Transport layer protocols, including ICMP, SNOS obfuscates are TCP (Transmission Control Protocol), UDP (User Datagram Protocol), and ICMP. An obfuscated ICMP packet is sent to the IP header obfuscation method and then to the next appropriate NDIS layer as Figure 23 shows.

For UDP and TCP packets, SNOS evaluates the packet to determine if an Application layer protocol needs obfuscation. Depending upon the Application layer protocol, SNOS distinguishes between client requests and server responses, as well as determining the specific response or request that is being passed. Incoming SMB Null Session request packets are dropped by SNOS because Nessus is able to determine that a host is running a Windows operating system by connecting to SMB using a Null Session connection. The SMB Null session packet is shown in Figure 24 by the "Yes" line originating from the "Incoming Null Request" decision and has an "X" on the line indicating that the packet is dropped.
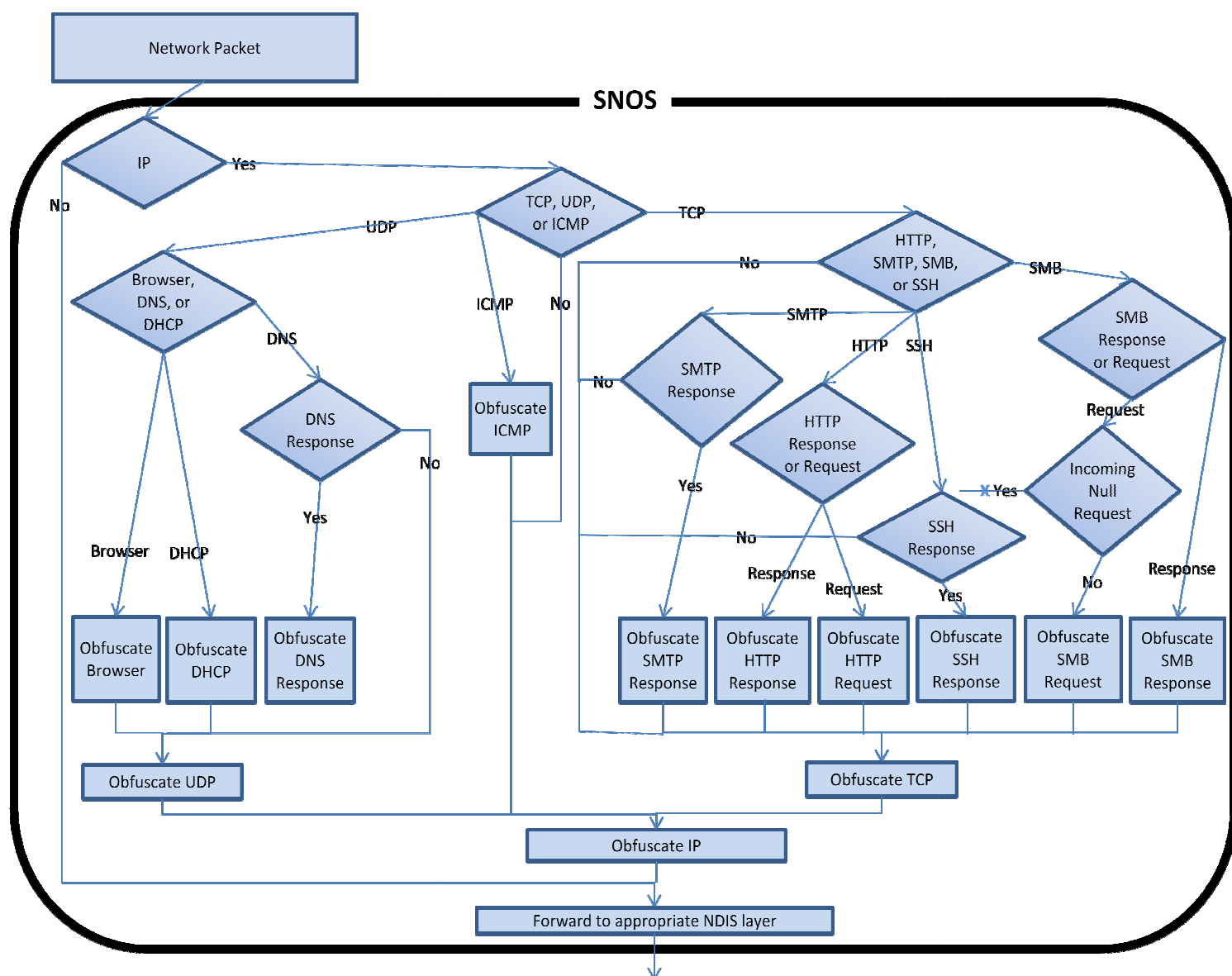
Figure 24: SNOS Packet Obfuscation

The remaining Application layer protocols are obfuscated appropriately and then, as shown in Figure 24, the packet is passed back down to the Transport layer protocol for obfuscation. The packet continues to be forwarded to the next lower TCP/IP model layer protocol for obfuscation until the packet is finished and forwarded to the correct NDIS layer – miniport or protocol. Appendix A contains the full list of protocols and modifications that SNOS performs to defeat Nmap and Nessus.

4.1.4 Host Functionality While Running SNOS

All four individual host configurations – email server, SharePoint server, web
server, and workstation, are tested to ensure that SNOS does not affect their functionality.
Functionality is defined as the ability for a service to operate correctly with a bare
minimum set of features enabled. For instance, a web server is considered fully
functional if web related traffic functioned normally. Integration of the selected
applications with other possible applications or configurations not specifically mentioned
are not tested.

Section 4.1.1 discusses the reasoning behind the protocol selection. These
protocols are used to determine the types of host configurations that are tested. Section
3.5.1 explains the statistics used to determine the specific applications used for each
protocol outlined in Section 4.1.1. The applications are installed on separate host devices
similar to production environments because email servers do not typically function as an
organizations web and SSH server. Likewise, workstations are not normally used to
provide email, file sharing, or web services. Applications and operating systems are
installed according to configuration guides from the vendor that created the application.
Guidelines and recommendations from the companies are followed when dividing up the
applications onto different hosts.

Only the protocols and services previously listed are tested. All other protocols
are blocked on each host. Windows firewall is configured on each host as part of the
base virtual image to only allow specific protocols to respond and receive connections
throughout the experiments. The email server primarily tests SMTP, DNS, and SMB
traffic, the SharePoint server tests SMB and HTTP response traffic (using an IIS web

68

server), the web server tests SMB and HTTP response traffic (using an Apache web server), and the workstation tests SMB, DHCP, SSH and HTTP request traffic (web surfing). All four hosts only test protocols mentioned in Section 4.1.1 and ignore other protocols inherently running on the host that are not directly used to provide functionality to the services being tested.

With SNOS enabled on each of the four host devices, specific traffic is passed to ensure that SNOS does not block or corrupt this network traffic. For the workstation, since web surfing encompasses the entire Internet and testing the full functionality of each website available on the Internet while running SNOS is not feasible, SNOS focuses on the top ten most visited internet sites [DFM06]. The workstation is considered fully functional for HTTP request traffic if all ten websites are functional while running SNOS.

SNOS purposefully blocks a portion of the SMB protocol's functionality. Windows, even Windows 7, operating systems allow incoming SMB Null sessions. To effectively obfuscate the host operating system, SMB Null sessions have to be blocked. SMB packets are still allowed and obfuscated by SNOS except for incoming SMB Null session requests. By completely blocking all incoming SMB Null sessions, SNOS also blocks additional information that can be combed from Null session queries – such as enumerating local logon credentials.

V. Results and Analysis

This chapter is divided into three sections. Section 5.1 shows the results of SNOS obfuscated packets. Section 5.2 shows the results from the obfuscation effectiveness experiment. Section 5.3 shows the results from the network latency experiment. The obfuscation effectiveness and network latency experiments' results are analyzed in Sections 5.4 and 5.5.

5.1   Packet Modification Results

The Exchange server results are primarily associated with SMTP, port 25. Since Exchange natively enables OWA (Outlook Web Access) by running an IIS web server, HTTP traffic is also used to determine if SNOS effectively modifies a packet originating from the Exchange server host.

The default SMTP banner produced by an Exchange server is shown in Figure 25. As discussed in Chapter 2, the default SMTP Exchange banner identifies the service – Microsoft ESMTP Mail Service – as a Microsoft Exchange server revealing the Windows operating system. The banner also identifies the exact version of Exchange (Exchange Server 2003) further suggesting that the host's operating system is Windows Server 2003. Figure 26 shows the SNOS-modified SMTP banner. The SNOS modified email banner identifies ARPANET as the email service application. ARPANET does not infer the host's operating system since no such email application exists by this name. SNOS can easily be implemented to replace the banner text with random words or other characters instead of using ARPANET in the SMTP banner.
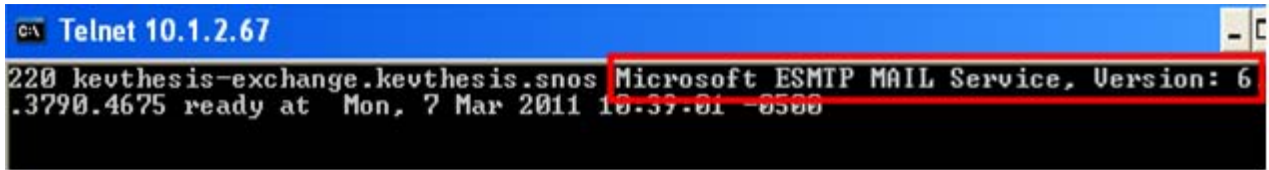
Figure 25: SMTP Banner without SNOS
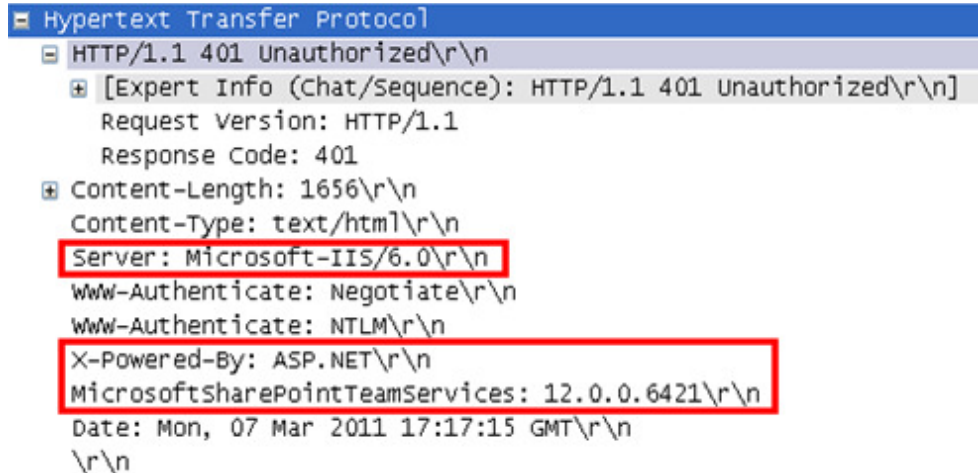
Figure 26: SMTP Banner Modified By SNOS

The naming convention used for the hosts is for the convenience of identifying a specific host during the experiments. Outside of this research environment, naming conventions should not contain information about the service or operating system running on the host. Naming convention problems are not a technical problem but a social problem within specific information technology departments. Good naming conventions should not infer any information about the hosts operating system or services.
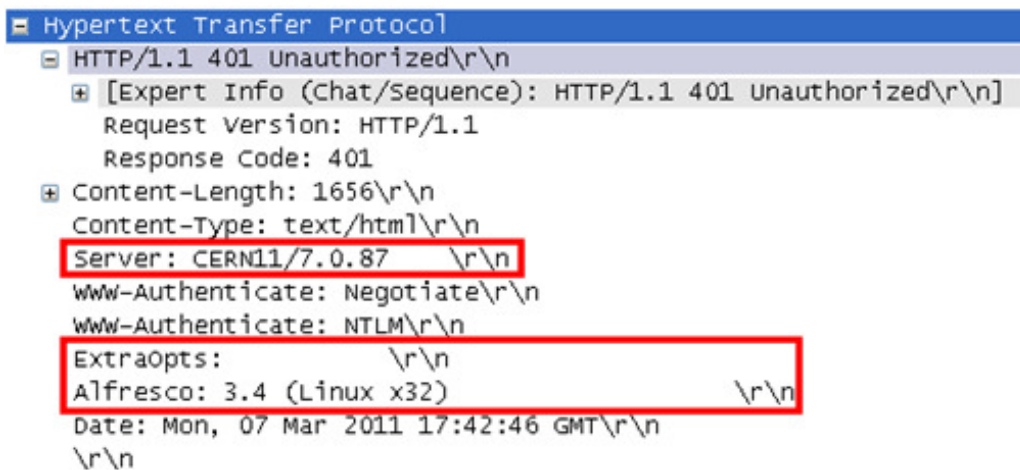
The standard SharePoint HTTP Response network packet is shown in Figure 27 compared with the SharePoint HTTP Response packet after SNOS has modified it, Figure 28. Without SNOS, the HTTP header identifies the web service being used – Microsoft-IIS/6.0, the programming language used – ASP.NET, and even explicitly identifies the server as running SharePoint Services. SNOS modifies all of the HTTP header fields, as shown in Figure 28 so that the web server type and, by inference, the host's operating system are obfuscated. The extra fields SNOS inserts into the HTTP header field are chosen without any particular preference and are inserted to replace the MicrosoftSharePointTeamServices field in Figure 27. The Server field identifies a CERN web server type. CERN was the first web server and ran on a UNIX operating

system and does not infer a Windows operating system. All SNOS-modified values can easily be programmed to use random characters or other preferred values.

Figure 27: SharePoint HTTP Header without SNOS

Figure 28: SharePoint HTTP Header Modified By SNOS

The web server's HTTP header is similar to SharePoint's HTTP header in Figure 27 and SNOS's modified version in Figure 28. The only difference is between the original HTTP Response packets is the Server, X-Powered-By and MicrosoftSharePointTeamServices fields. Figure 29 shows the standard HTTP response from the web server without SNOS's modifications. The server field identifies the Windows operating system. The SNOS program modified the Server field for the web

server identical to the SharePoint server's server field – CERN11/7.0.87 – shown in

Figure 28.

Figure 29: Web HTTP Header without SNOS

The User-Agent HTTP Request header field identifies the web browser type and

version. The unmodified Internet Explorer created HTTP Request packet is shown in

Figure 30 - the User-Agent field identifies the host's operating system as a Windows

operating system using Internet Explorer 8. The SNOS modified HTTP request packet is

shown in Figure 31. The User-Agent field is modified to resemble a Mozilla Firefox web

browser running on a Linux operating system.

Figure 30: HTTP Request without SNOS

Figure 31: HTTP Request Modified By SNOS

Effective operating system obfuscation requires ICMP packet obfuscation. ICMP packets are highly variable between operating systems. Figure 32 is a standard Windows ICMP echo reply packet. To obfuscate the ICMP packet, as shown in Figure 33, SNOS creates a random Identifier number, changes the length of the ICMP payload from 32 bytes to 56 bytes, and changes the payload characters from the Windows default alphabet to random characters.
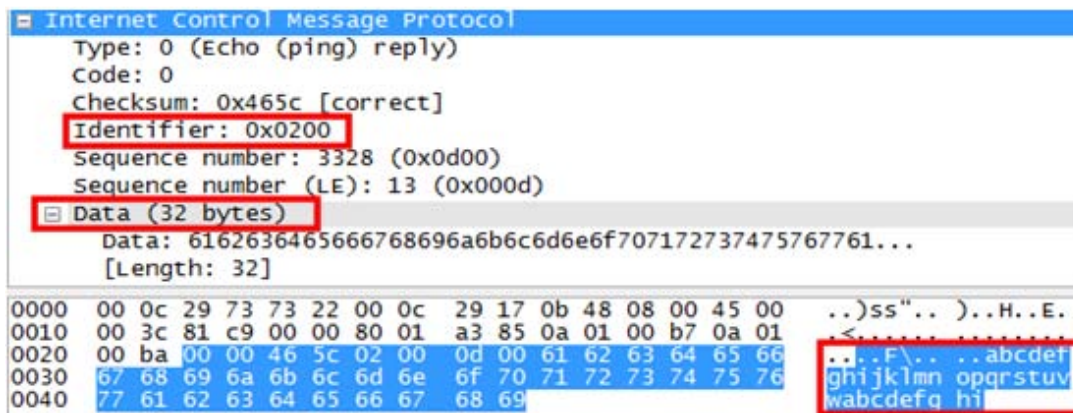
Figure 32: ICMP without SNOS
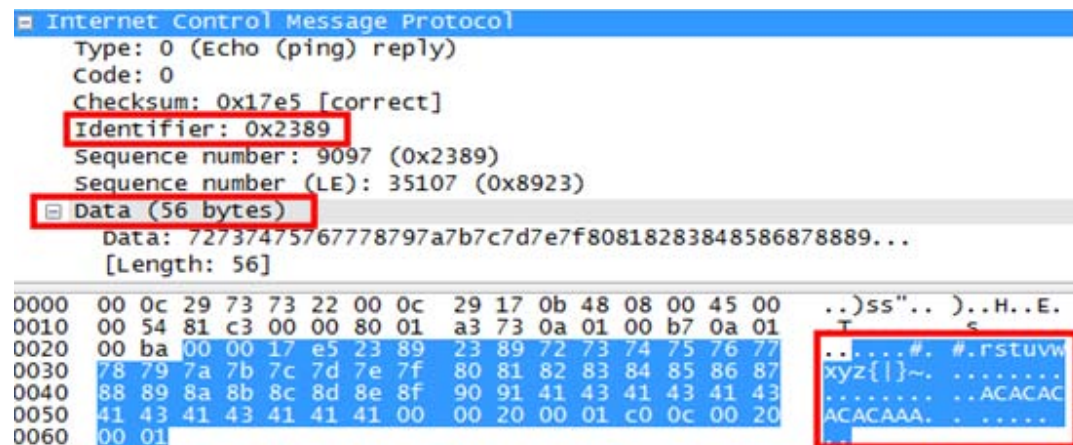
Figure 33: ICMP Modified By SNOS

Windows XP and Windows Server 2003 operating systems enable the SMB protocol by default, so the SMB protocol is tested, and obfuscated on each of the four host devices. SMB Response packets – particularly the Session Setup AndX Response

packet – identify the host operating system and the service providing SMB. Figure 34

shows the default Windows SMB response packet – identifying the operating system as

either Windows XP or Windows Server 2003. The Native OS and Native Lan Manager

fields inside a SMB AndX Response packet identify the host's operating system. SNOS

obfuscates the Native OS and Native Lan Manager fields, shown in Figure 35, to imitate

a Linux operating system running Samba. Samba can be installed on a Linux operating

system to allow a Linux host to share and communicate with the Windows SMB service.

The flags are also changed by SNOS.


```
☐ SMB (Server Message Block Protocol)
   ☐ SMB Header
        Server Component: SMB
        [Response to: 1075]
        [Time from request: 0.000214000 seconds]
        SMB Command: Session Setup AndX (0x73)
        NT Status: STATUS_MORE_PROCESSING_REQUIRED (0xc0000016)
      ⊞ Flags: 0x98
      ⊞ Flags2: 0xc807
        Process ID High: 0
        Signature: 0000000000000000
        Reserved: 0000
        Tree ID: 0
        Process ID: 65279
        User ID: 10241
        Multiplex ID: 16
   ☐ Session Setup AndX Response (0x73)
        Word Count (WCT): 4
        AndXCommand: No further commands (0xff)
        Reserved: 00
        AndXOffset: 331
      ⊞ Action: 0x0000
        Security Blob Length: 214
        Byte Count (BCC): 288
      ⊞ Security Blob: 4e544c4d53535000020000001a001a0038000000
        Native OS: Windows 5.1
        Native LAN Manager: Windows 2000 LAN Manager
```

Figure 34: SMB Response without SNOS
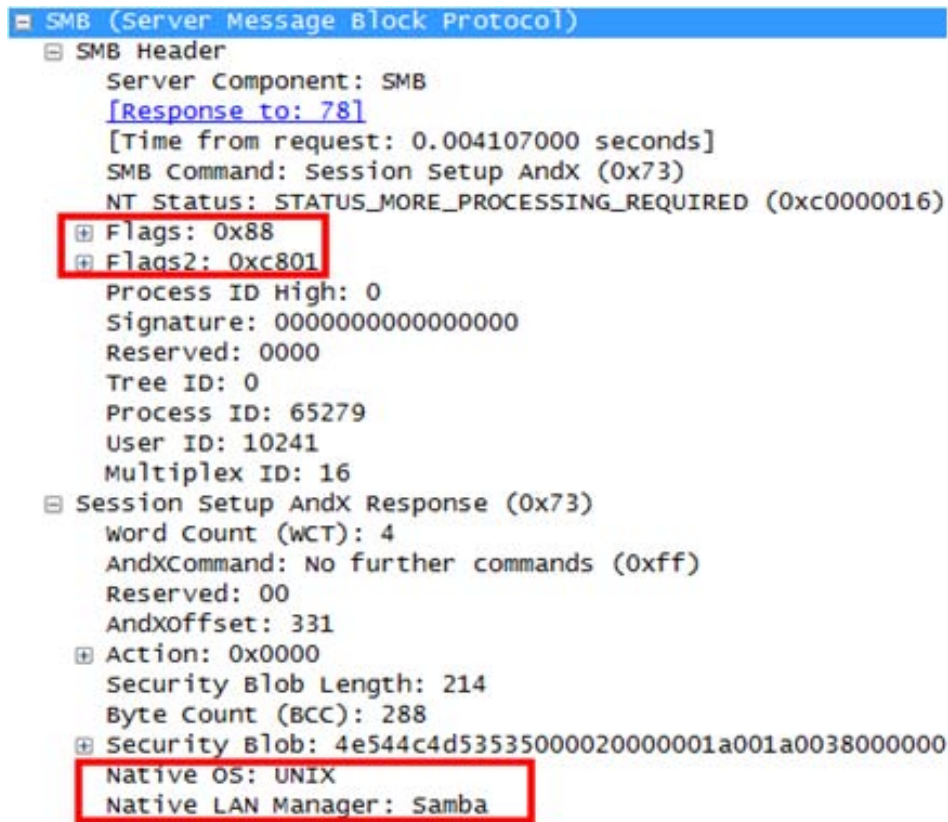
Figure 35: SMB Response Modified By SNOS

A complete list of SNOS obfuscation modifications is shown in Appendix A. This list includes additional obfuscation not specifically referenced in this section.

5.2   Obfuscation Effectiveness Results

The obfuscation effectiveness experiment tests the obfuscation effectiveness metric. The results of this experiment determine the success or failure of SNOS in obfuscating the Windows operating system. The obfuscation effectiveness experiment consists of three separate trials - each repeated forty times, and each trial consists of four separate tests, as explained in Section 3.6. The baseline trial is used to compare the obfuscation effectiveness of the two programs tested – OSfuscate and SNOS. After forty experimental runs, the experiment yields 1,920 individual results corresponding to a

specific fingerprinting test against a specific host. Appendix B contains the results of the

forty experimental runs. The results are detailed in the following sections and are

initially divided into the four tests – Nmap OS Class, Nmap Service, Nessus OS Class,

and Nessus Service.

5.2.1   Nmap OS Class Test

The Nmap OS Class test is the process Nmap uses to directly detect a host's

operating system. The Nmap OS Class test is run against each host during each trial.

Figure 36 is a screenshot from the Nmap OS Class test for the benchmark trial.

Figure 36 shows the ports the Nmap OS Class test used to identify the host and a list of

possible operating systems. The benchmark trial always yields an accurate operating

system fingerprint – Windows XP or Windows Server 2003. Figure 36 represents the

results of the Nmap OS Class test for the Exchange server during the benchmark trial.

The other three hosts return identical results with the only difference being the port

numbers the Nmap OS Class uses to fingerprint the operating system.
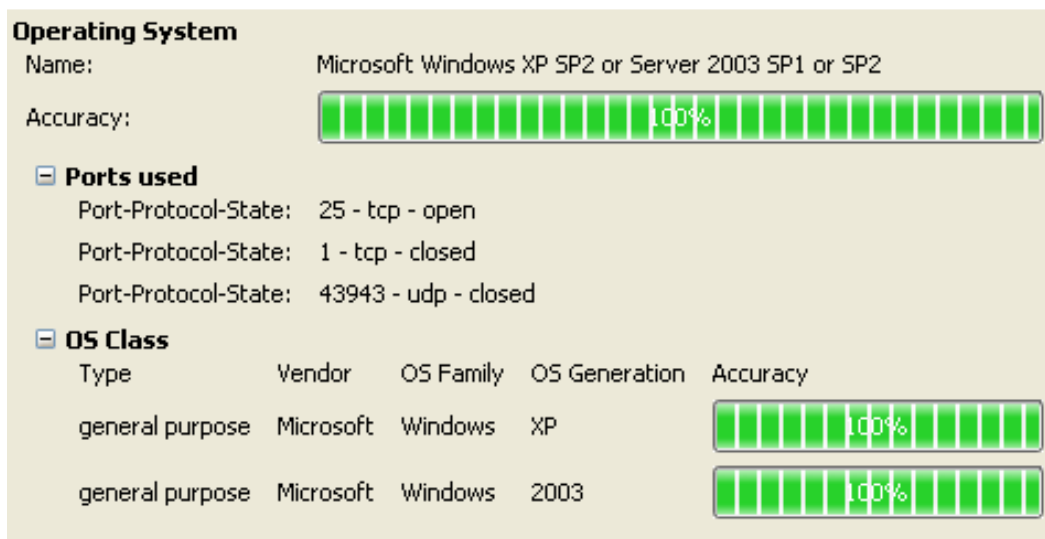
Figure 36: Nmap OS Class – Benchmark Trial

77

During the benchmark trial, the Nmap OS Class confidence level is 100% for thirty-eight out of the forty repetitions. One result for the SharePoint host is 99% confident and another result for the workstation host is 98% confident. In both cases, the Nmap OS Class only identifies Windows as the possible operating system matching the fingerprint.

The OSfuscate trial produces similar results as the benchmark. The Nmap OS Class test identifies the Windows operating system every time by using the same port numbers for each host as the benchmark trial. The difference between the benchmark and the OSfuscate trials is that Nmap OS Class is less confident that the fingerprint result is accurate. Nmap suggests additional operating systems as a possible fingerprint match besides the Windows operating system. This difference is inconsequential because the Nmap OS Class test still identifies the Windows operating system every time. The confidence level of Nmap for the OS Class test remains over 85% for all results produced during the OSfuscate trial.

Figure 37 shows a result from the OSfuscate trial and how Nmap associates additional operating systems to the fingerprint. These devices include switches, printers, and routers. Since the Nmap OS Class test is no longer 100% confident with the results accuracy, Nmap displays a comprehensive list of possible operating systems that could be running on the host. This result can be considered a form of obfuscation. Several times the additional operating systems, such as the Dell PowerConnect operating system, are listed with a higher confidence interval, by 1% or 2% more than Windows. The purpose of this research is not to determine the confidence level of Nmap or how that confidence level might be interpreted by a wide range of potential users. If Windows is listed as a

result of the Nmap OS Class test then the test is considered successful at identifying the

host's operating system.
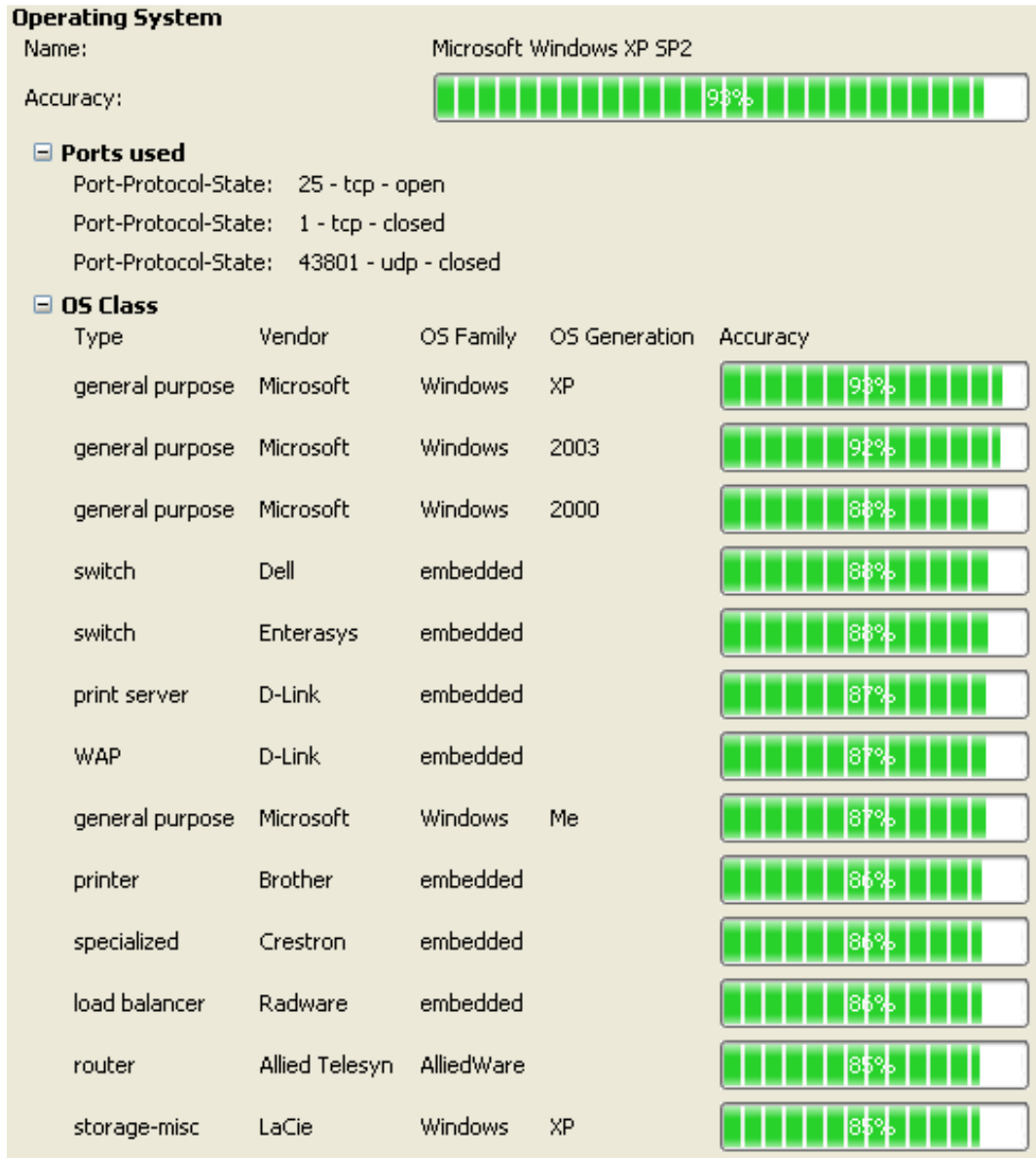
Figure 37: Nmap OS Class – OSfuscate Trial

The Nmap OS Class is never able to accurately identify the host's operating

system for any of the four hosts during the SNOS trial. The Nmap OS Class test uses the

same port as the benchmark trial to fingerprint the host for all four hosts. The Nmap OS

Class test results shows that the operating system is a Dell PowerConnect management

switch – as shown in Figure 38, a firewall or at times failed to provide a guess. The

Nmap OS Class produces an 80-90% confidence level when identifying the host as a Dell

PowerConnect or firewall operating system.

Unlike the OSfuscate trial, the Nmap OS Class test results for the SNOS trial

never list Windows as a possible operating system matching the fingerprint. SNOS

successfully obfuscates the host operating system such that the Nmap OS Class test no

longer lists the Windows operating system. Whether or not a wrong result is better or

worse than no result is not examined further because the significance of this difference

depends upon individual user evaluation. Additional studies and discussions would need

to address which result would be more favorable in order to obfuscate the operating

system from the user. Results identified are all or nothing regarding success in

obfuscating the Windows operating system.
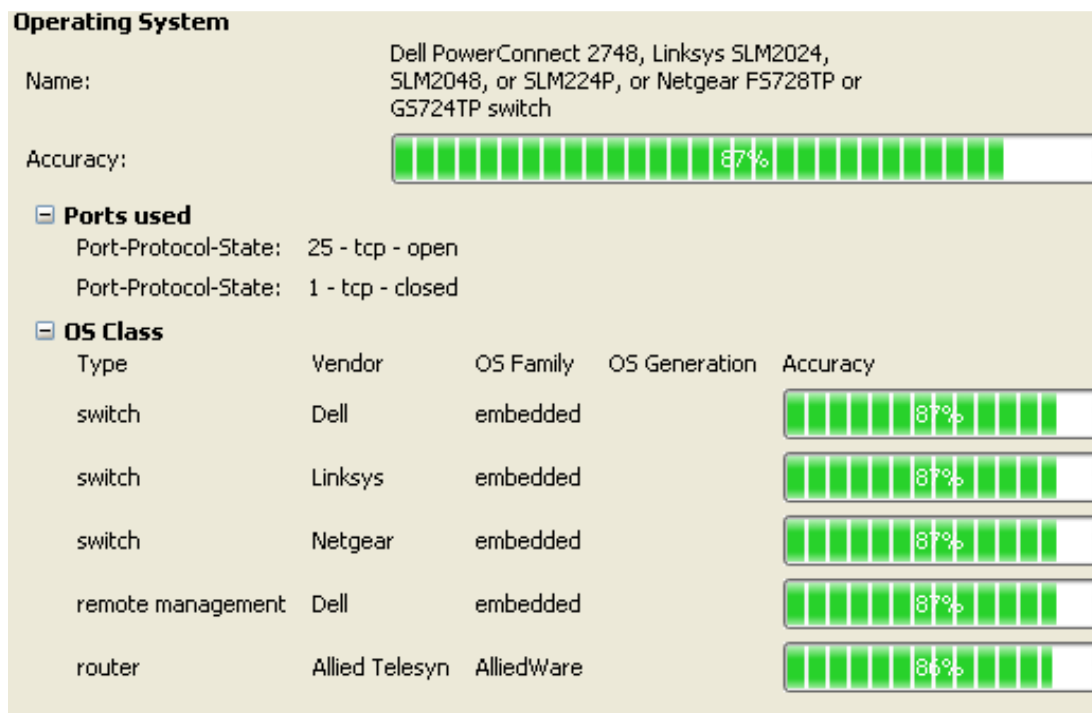
Figure 38: Nmap OS Class – SNOS Trial

Under the benchmark trial – "No Obfuscation" – shown in Figure 39, the Nmap

OS Class test is able to successfully identify the operating system for each host. The

individual cells under the "No Obfuscation" column indicate if the Nmap OS Class is

successful with either a Y for yes or N for no, the port number Nmap OS Class uses to

fingerprint the operating system, and the confidence percentage. Nmap is at least 98%

confident in the result of the Nmap OS Class fingerprint throughout all forty

experimental runs for the "No Obfuscation," benchmark, trial. Appendix B contains the

full results from all forty experimental runs.

SNOS successfully obfuscates the host's operating system as noted by the "N"

under the SNOS column in Figure 39. The port number after the "N" indicates the port

number that the Nmap OS Class test uses to fingerprint the operating system. The

percentage shows the confidence level for the listed operating system that Nmap matches

to the fingerprint. The SharePoint cell under the SNOS column has no port number or

confidence level because Nmap OS Class could not match the results to a known

fingerprint signature in Nmap's database.

Run #: 38

	No Obfuscation Nmap OS Class	OSfuscate Nmap OS Class	SNOS Nmap OS Class
Exchange	Y(25)100%	Y(25)90%	N(25)85%
Sharepoint	Y(80)100%	Y(80)89%	N
Web	Y(80)100%	Y(80)90%	N(80)85%
Workstation	Y(22)100%	Y(22)89%	N(22)91%

Figure 39: Nmap OS Class Experimental Run

Appendix B contains the full results recorded from the Nmap OS Class test against each host. Figure 36, 37, and 38 are taken from experimental run thirty-eight and represent the row labeled Exchange as identified in Figure 39.

5.2.2   Nmap Service Test

The Nmap Service test uses Nmap's ability to fingerprint a host's services. The Nmap Service test is considered as significant as the Nmap OS Class test in determining the effectiveness of SNOS relative to the benchmark and compared against OSfuscate because services, directly or through inference, identify the host's operating system.

Figure 40 shows the results from the Nmap Service test performed against the Exchange server host. Figure 40 only shows the results for protocols previously selected for obfuscation. Additional identified protocols are ignored as part of the result returned from the Nmap Service test.

The result of each identified service is considered successful in identifying the operating system if, under the Version column, the Nmap Service test identifies the operating system. In Figure 40, Nmap Service identifies port 25 as running Microsoft ESMTP – which is Microsoft Exchange; therefore inferring the Windows operating system. Port 80 allows for a similar inference based on the Version information identified as Microsoft IIS. Port 445, SMB, produces the most concise information by specifically identifying the operating system as being Windows Server 2003 or Windows Server 2008.

Figure 40: Nmap Service – Benchmark

The Nmap Service test produces the exact same result when run during the

OSfuscate trial for each host.   Figure 41 represents the results returned during the

OSfuscate trial with no differences reported throughout the entire experiment – all forty

experimental repetitions on all four hosts.

Figure 41: Nmap Service – OSfuscate

For both the benchmark and OSfuscate trials against the Workstation host, Nmap

Service occasionally does not identify port 445's version information, as shown in Figure

42. During the benchmark trial for the Nmap Service test, the workstation's operating

system is not identified 37% of the time (15 out of 40) and during the OSfucate trial, the

workstation's operating system is not identified 35% of the time (14 out of 40). The

inability of Nmap Service to indentify port 445's version information is further discussed

in Section 5.4.1 as part of the analysis of the Nmap results.

Run #: 14

	No Obfuscation	OSfuscate	SNOS
	Nmap	**Nmap**	**Nmap**
	Services	*Services*	*Services*
Exchange	Y(25,80,445)	Y(25,53,80,445)	N
Sharepoint	Y(80,445)	Y(80,445)	N
Web	Y(80,445)	Y(80,445)	N
Workstation	N	N	N

Figure 42: Nmap Service – Workstation SMB Not Identified

The results from the Nmap Service test for the Exchange server host during the

SNOS trial are shown in Figure 43. The Nmap Service is unable to identify any of the

obfuscated protocols as shown by the empty Version column in Figure 43.

Nmap Output	Ports / Hosts	Topology	Host Details	Scans

Port	Protocol	State	Service	Version
25	tcp	open	smtp	
53	tcp	open	domain	
80	tcp	open	http	
139	tcp	open	netbios-ssn	
389	tcp	open	ldap	
445	tcp	open	microsoft-ds	

Figure 43: Nmap Service – SNOS

The Nmap Service test results from all three trials are shown in Appendix B under

the Nmap Service column. Figure 40, 41, and 43 represent the data used to populate the

Exchange row in Figure 44. Each cell in Figure 44 has a "Y" if Nmap Service identifies

the Windows operating system or an "N" if Nmap Service does not identify the Windows

operating system. Except for the occasional lack of Version information for port 445 on

the workstation, the protocols used to identify the host's operating system are identical

between the benchmark and OSfuscate trials. These trials result in the Nmap Service test

correctly identifying the underlying operating system 135 out of 160 times and 136 out of

160 times for the benchmark and OSfuscate trials, respectively. The SNOS trial results in

the Nmap Service test never being able to identify the correct operating system during all

forty repetitions on all four hosts or 0 out of 160 results.

Run #: 19

	No Obfuscation	OSfuscate	SNOS
	Nmap Services	Nmap Services	Nmap Services
Exchange	Y(25,80,445)	Y(25,53,80,445)	N
Sharepoint	Y(80,445)	Y(80,445)	N
Web	Y(80,445)	Y(80,445)	N
Workstation	Y(445)	Y(445)	N

Figure 44: Nmap Service Experimental Run

5.2.3   *Nessus OS Class Test*

Similar to Nmap, Nessus uses multiple operating system and service

fingerprinting techniques. The Nessus OS Class test is also referred to as Nessus OS

Identification. The Nessus OS Class test results are found inside the 0/tcp subsection of a

Nessus report.

The Nessus report does not detail the ports used to fingerprint the operating

system. Instead, the Nessus OS Class test uses the techniques discussed in Section 2.4.2.

The Nessus OS Class results from the SNOS trial provide information about how Nessus

uses operating system fingerprinting techniques. The SNOS trial does not produce an

operating system match, so Nessus displays the fingerprint created from the test.

Figure 45 shows the result of one of the benchmark trial runs against the

Exchange server. Nessus is able to accurately detect the exact type and version of the

operating system running. The confidence level of 99 shown in Figure 45 remains

constant every time Nessus OS Class is able to correctly fingerprint the operating system

for the benchmark and OSfuscate trials.

The Nessus OS Class test reports the method as MSRPC used to fingerprint the

operating system but does not completely identify the techniques used. Even when all

MSRPC related ports are blocked by Windows firewall on the host, Nessus still reports

the method as MSRPC because of ports 139 and 445 which are used by SMB.
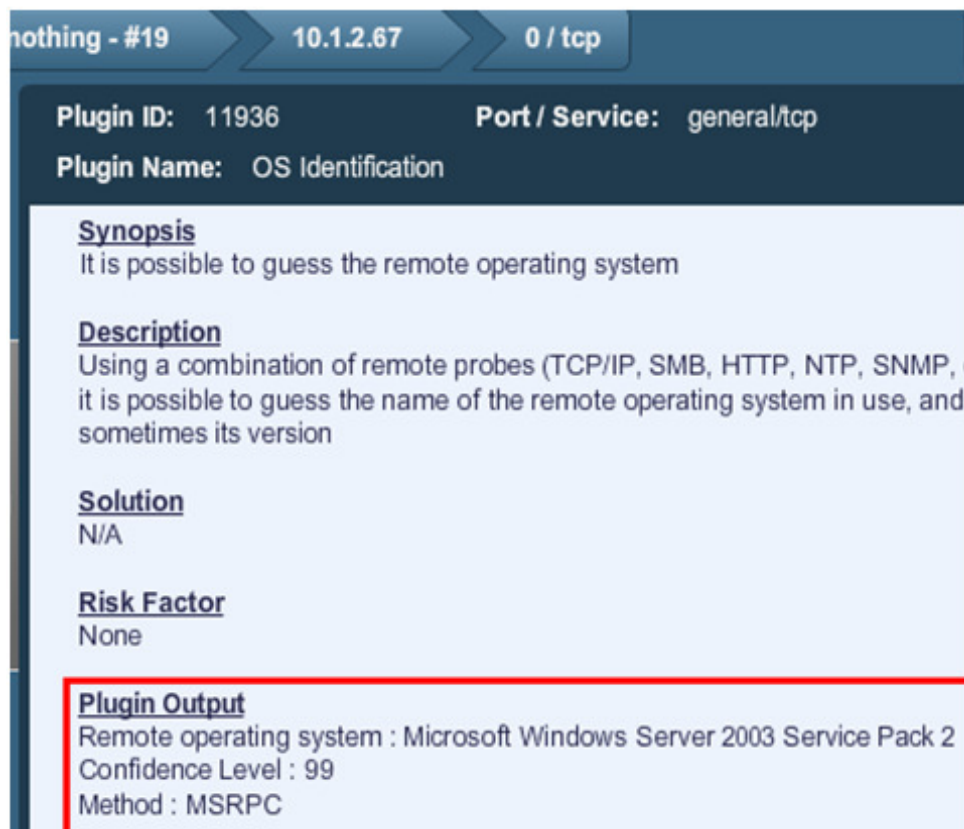
Figure 45: Nessus OS Class - Benchmark

An OSfuscate trial result is shown in Figure 46. The results from the OSfuscate

and the benchmark trials are identical as shown by comparing Figure 45 with Figure 46.

Nessus OS Class is able to accurately fingerprint the operating system's type and version.

These results are identical for all four hosts during all forty repetitions of the benchmark

and OSfuscate trials.
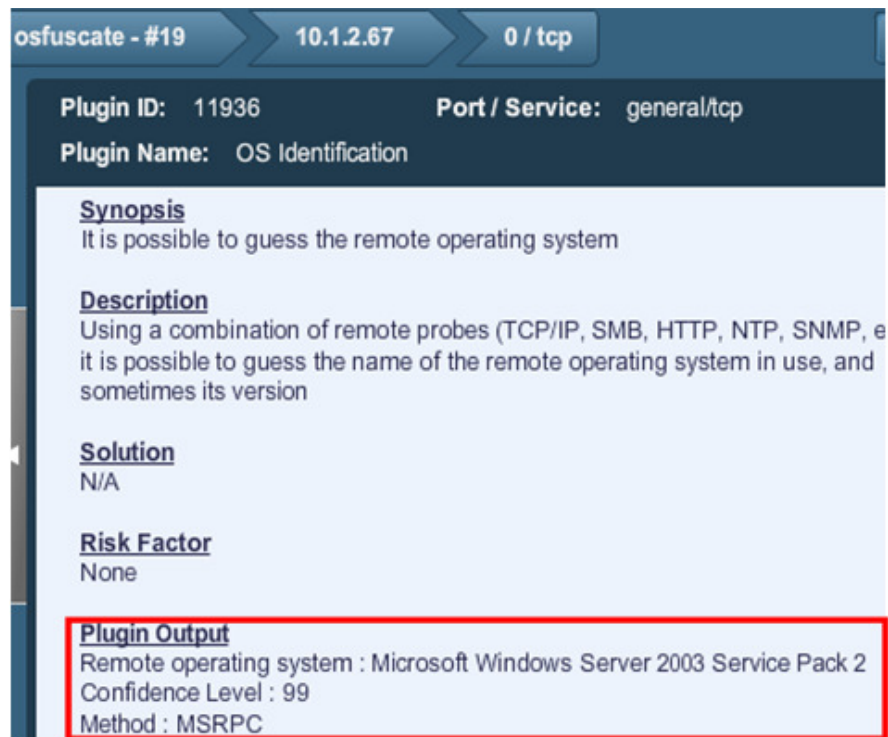
Figure 46: Nessus OS Class – OSfuscate

A SNOS trial result, Figure 47, shows that Nessus OS Class fails to identify the operating system of the host. Under the Synopsis section, Nessus OS Class states, "it was not possible to guess the remote system." The plug-in output in Figure 47 shows the methods Nessus OS Class tries to use to identify the operating system. The HTTP output displays the server field from the HTTP response packet. The SinFP output is the fingerprint created by the SinFP algorithm that Nessus cannot match to an operating system fingerprint within Nessus' database.
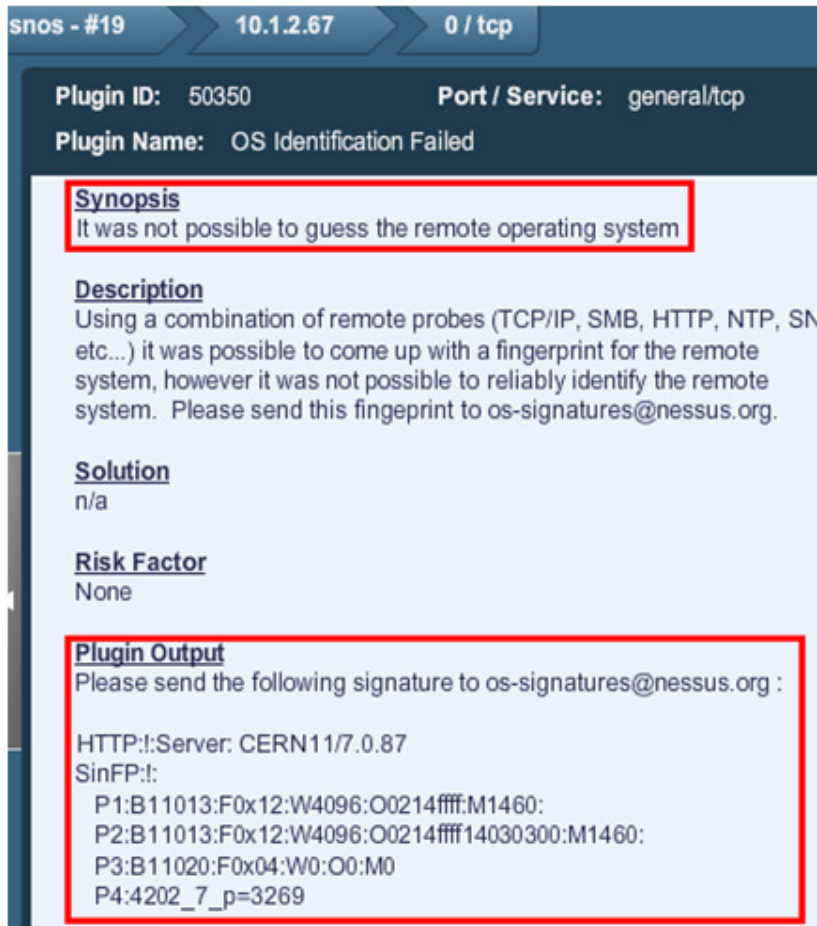
Figure 47: Nessus OS Class – SNOS

Figure 48 shows experimental run, repetition, number nineteen. Figure 45, 46, and 47 are represented by the Exchange server row in Figure 48. The results produced within each trial throughout the forty experimental runs are identical. Nessus OS Class is able to correctly identify the operating system 100% of the time for the benchmark and OSfuscate trials and 0% of the time for the SNOS trial. The Y, yes, in Figure 48 records that Nessus OS Class correctly identifies the operating system, and the N, no, means that Nessus does not identify the Windows operating system. The protocols listed in the cells represent the protocols Nessus OS Class uses to fingerprint the operating system. The complete results of all forty repetitions are attached as Appendix B.

Run #: 19

	No Obfuscation	OSfuscate	SNOS
	Nessus	Nessus	Nessus
	OS Ident	OS Ident	OS Ident
Exchange	Y(TCP/IP, SMB, HTTP)	Y(TCP/IP, SMB, HTTP)	N
Sharepoint	Y(TCP/IP, SMB, HTTP)	Y(TCP/IP, SMB, HTTP)	N
Web	Y(TCP/IP, SMB, HTTP)	Y(TCP/IP, SMB, HTTP)	N
Workstation	Y(TCP/IP, SMB)	Y(TCP/IP, SMB)	N

Figure 48: Nessus OS Class Experimental Run
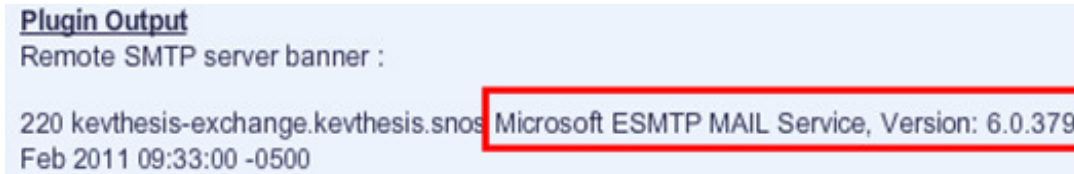
5.2.4   Nessus Service Test

The Nessus Service test results are not as easily viewable as the Nmap Service test results. The Nmap Service test lists all the detected services and any identified information about each service under the Ports / Hosts tab. The report generated by Nessus requires selecting each detected protocol and then viewing the protocols details to determine if the Nessus Service test identifies the service and/or operating system. Figure 49 shows how Nessus lists the detected services. Only the services relating to the protocols selected for this research are shown and used as a means of detecting the underlying operating system.

nothing - #19		10.1.2.67	
Port ▲	Protocol	SVC Name	Total
0	udp	general	1
0	tcp	general	8
0	icmp	general	1
25	tcp	smtp	4
80	tcp	www	7
139	tcp	smb	3
389	tcp	ldap	6
445	tcp	cifs	13

Figure 49: Nessus Service –Service List
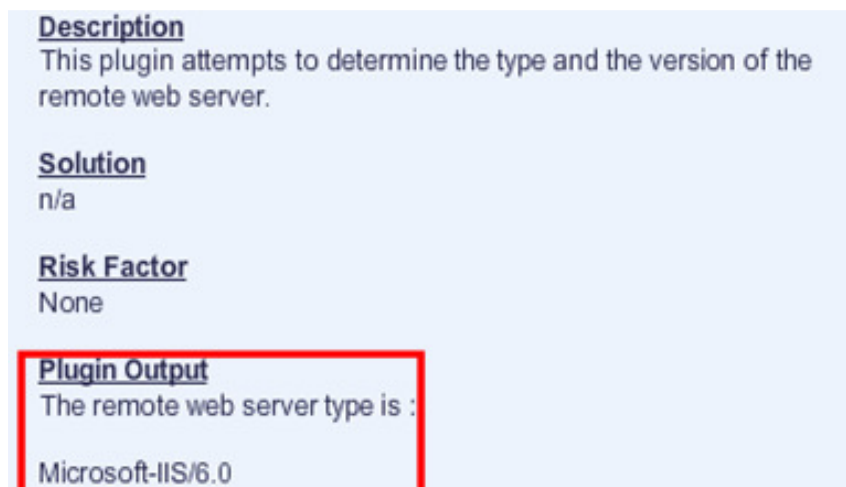
The Exchange server host uses SMTP, HTTP and SMB as shown in Figure 49. The benchmark trial correctly identifies the server type and version running SMTP by looking at the SMTP banner as shown in Figure 50.

Plugin Output
Remote SMTP server banner :

220 kevthesis-exchange.kevthesis.snos Microsoft ESMTP MAIL Service, Version: 6.0.379
Feb 2011 09:33:00 -0500

Figure 50: Nessus Service – Benchmark - SMTP

Nessus Service identifies the HTTP application running by looking at the server field inside of the HTTP response packet shown in Figure 51. The server field identifies the Microsoft IIS web server which infers the host is using a Windows operating system. Nessus also displays the entire HTTP Response packet header which identifies the web application server running by viewing the Content-Location field and X-Powered-By fields as shown in Figure 52. The Content-Location field identifies the iisstart.htm default page which is the default webpage for the Microsoft IIS web server. The X-Powered-By field indicates the programming/scripting language used by the web server as ASP.NET which natively runs on a Microsoft IIS web server.

Description
This plugin attempts to determine the type and the version of the remote web server.

Solution
n/a

Risk Factor
None

Plugin Output
The remote web server type is :

Microsoft-IIS/6.0

Figure 51: Nessus Service – Benchmark - HTTP

Figure 52: Nessus Service – Benchmark - HTTP Response Header

The SharePoint server HTTP Response header contains additional fields as shown

in Figure 53. The HTTP Response header identifies that the server is running Microsoft

SharePoint Services. The Nessus Service test creates a separate item, similar to the

Server field in the HTTP Response header that identifies Microsoft SharePoint Services

running on the host.

Figure 53: Nessus Service – Benchmark – SharePoint HTTP Header Response

A Nessus Service result for the SMB protocol is shown in Figure 54. Figure 54 is the AndX Session Setup SMB command sent back from a host. The AndX Session Setup SMB packet identifies the operating system using the Native OS, and Native Lan Manager fields. Using this packet, Nessus Service accurately identifies the exact type and version of the operating system running – Windows Server 2003 Service Pack 2.

Plugin Output
The remote Operating System is : Windows Server 2003 3790 Service Pack 2
The remote native lan manager is : Windows Server 2003 5.2
The remote SMB Domain Name is : KEVTHESIS-SHARE

Figure 54: Nessus Service – Benchmark – SMB

Nessus Service attempts to connect to the detected SMB port using a Null Session request. If Nessus is able to connect with a Null Session, then Nessus Service identifies the operating system as a Windows operating system, shown in Figure 55.

Plugin Name: Windows SMB NULL Session Authentication

Synopsis
It is possible to log into the remote Windows host with a NULL session.

Description
The remote host is running Microsoft Windows, and it was possible to log into it using a NULL session (i.e., with no login or password). An unauthenticated remote attacker can leverage this issue to get information about the remote host.

Figure 55: Nessus Service – Benchmark – SMB Null Session

The Nessus Service results during the OSfuscate trial are identical to the benchmark trial for each service on all four hosts. OSfuscate does not obfuscate any information regarding any of the services fingerprinted by the Nessus Service test.

The SNOS trial produces successful results in defeating the Nessus Service test for each protocol. Nessus grabs the SMTP banner in the SNOS trial, Figure 56, but the

SMTP banner does not infer the Windows operating system because the banner provides

false information regarding the SMTP service running.

Plugin Output
Remote SMTP server banner :

220 kevthesis-exchange.kevthesis.snos ARPANET2 8.12.9/8.12.9

Figure 56: Nessus Service – SNOS – SMTP

The HTTP service detected by Nessus does not identify the correct service type

and version running and does not give information inferring the Windows operating

system. Figure 57 shows the HTTP service information that Nessus reported.

Plugin Output
The remote web server type is :

CERN11/7.0.87

Figure 57: Nessus Service – SNOS - HTTP

The HTTP response header that Nessus reports is obfuscated by SNOS in Figure

58. The Content-Location field is modified and other fields, such as the X-Powered-By

field and the MicrosoftSharepointServices field for the SharePoint server results are

removed from the HTTP response header. The Content-Location field no longer displays

the default iisstart.htm value.

Content-Length: 1433
Content-Type: text/html
Content-Location: http://10.1.2.67/thankyou.xml
Last-Modified: Fri, 21 Feb 2003 23:48:30 GMT
Accept-Ranges: bytes
ETag: "09b60bc3dac21:623"
Server: CERN11/7.0.87
ExtraOpts:
Date: Mon, 21 Feb 2011 14:12:47 GMT
Connection: close

Figure 58: Nessus Service – SNOS – HTTP Response Header

SNOS is successful at obfuscating the SMB protocol from Nessus, Figure 59. SNOS obfuscates the Native OS, and Native Lan Manager fields, so these fields no longer disclose the Windows operating system. SNOS blocks incoming SMB Null session requests, which does not allow Nessus to determine the operating system by connecting to a SMB Null session.

Plugin Output
The remote Operating System is : Unix
The remote native lan manager is :
The remote SMB Domain Name is : SNOS

Figure 59: Nessus Service – SNOS – SMB

The results in Figure 50 through Figure 59, except Figure 53, are summarized in Figure 60 and represent the results recorded in the Exchange row. Figure 53 represents a cell in the SharePoint server row under the "No Obfuscation" column. The complete list of results for all forty repetitions is shown in Appendix B.

Run #: 19

	No Obfuscation Nessus Services	OSfuscate Nessus Services	SNOS Nessus Services
Exchange	Y(25,80,445)	Y(25,80,445)	N(445)
Sharepoint	Y(80,445)	Y(80,445)	N(445)
Web	Y(80,445)	Y(80,445)	N(445)
Workstation	Y(445)	Y(445)	N(445)

Figure 60: Nessus Service Experimental Run

5.3 Network Latency Results

The Network Latency experiment consists of two trials – the benchmark and the SNOS trials. Appendix C contains the results from all ten experimental runs for the Network Latency experiment. Network latency is measured by using the roundtrip time.

Figure 61 shows experimental run #10 taken from Appendix C. Each experimental run produces twelve results per host. Each trial in the network latency experimental ran for one hour collecting the twelve results. A mean is calculated for the roundtrip time every five minutes throughout the hour from the results obtained during that specific five minute interval. This five minute mean is used as a single result – one of the twelve – during the one hour trial run. A five minute average roundtrip time as the result help eliminate the variance that a rogue packet can have over the course of the entire experiment's results. A rogue packet is a packet that for some reason is delayed or lost in transit. Rogue packets can occur because of collisions at the NIC, on the wire, or corruption of the packet throughout the transmission process.

The Mean columns for each trial shown in Figure 61 represent the average roundtrip time of the twelve results for each host. The Min columns represent the minimum roundtrip time out of the twelve results for each host. The Max columns represent the maximum roundtrip time out of the twelve results for each host. The SD columns are the standard deviations for each set of twelve results per host and the CI columns represent the 95% confidence interval for each host using the twelve results. The numbers represented in each cell are recorded in microseconds to more accurately detect any differences between the trials. The network latency analysis, Section 5.5, uses the combined results for all ten repetitions to calculate the experimental mean, confidence interval, comparison of means, box plots, and ANOVA (Analysis of Variance) tables for each host.

Run #: 10

	No Obfuscation					SNOS				
	Min	Max	Mean	SD	CI	Min	Max	Mean	SD	CI
Exchange	195.00	347.00	235.42	39.78	29.58	203.00	328.00	237.50	39.11	29.08
Sharepoint	244.00	422.00	294.25	52.40	38.96	271.00	385.00	322.92	30.82	22.92
Web	236.00	305.00	267.58	23.39	17.39	273.00	949.00	428.83	214.07	159.18
Workstation	250.00	1180.00	404.58	301.89	224.48	327.00	483.00	376.58	48.77	36.26

Figure 61: Network Latency Result

5.4   Obfuscation Effectiveness Analysis

The obfuscation effectiveness experiment yields 1,920 individual results – 40 experimental runs x 3 trials x 4 fingerprinting tests x 4 hosts. Figure 62 shows a histogram of all 1,920 results separated by trial. The variance in both the benchmark and OSfuscate trials are discussed in more detail in Section 5.4.1 and resulted from the Nmap Service test against the workstation host.   Appendices D and F list all 1,920 results returned throughout the obfuscation effectiveness experiment.
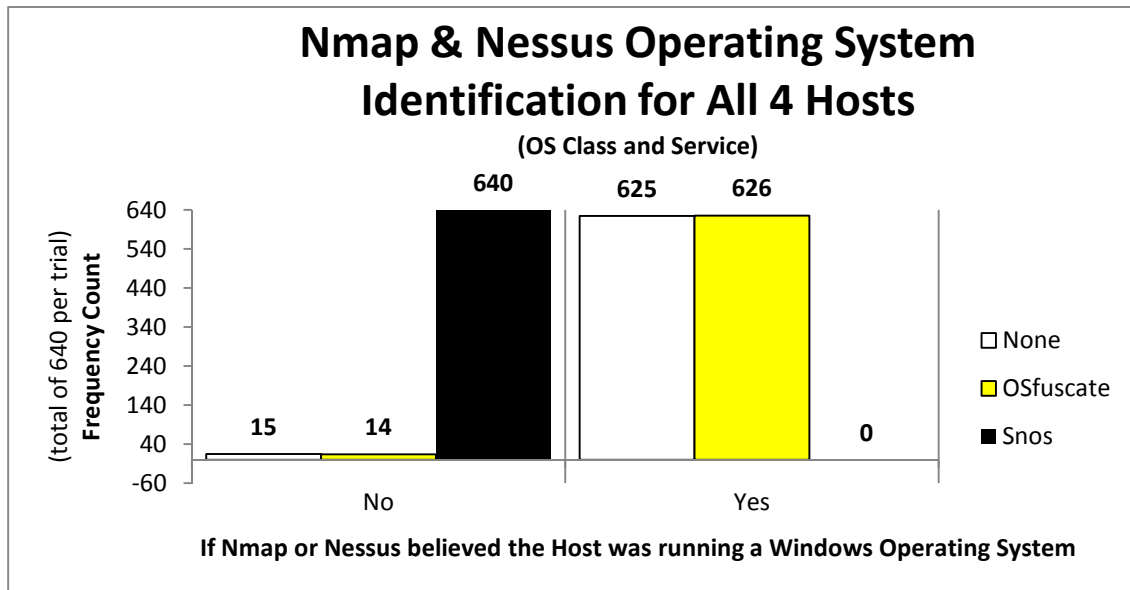
Figure 62: Nmap and Nessus Obfuscation Results for All Hosts and Tests

A more accurate depiction of the effectiveness of OSfuscate and SNOS compared with the benchmark trial is viewed when grouping the two tests – OS Class and Service – for each fingerprinting program into a combined test result. The analysis of the results from the obfuscation effectiveness experiment are grouped together according to the fingerprinting program used – Nmap or Nessus.

The analysis is grouped together because when either program runs (using the parameters discussed in Section 2.4.1 for Nmap and Section 2.4.2 for Nessus) the OS Class and Service tests are simultaneously run against the host. The results are recorded separately because each test is critical to determine the effectiveness of the SNOS obfuscation process relative to OSfuscate and the benchmark. The results show that OSfuscate focuses on obfuscating the Nmap OS Class test and does not obfuscate the remaining tests. By grouping the two tests for each fingerprinting program together, the analysis shows that both tests must be defeated to successfully obfuscate the host's operating system because both tests automatically run and if one test identifies Windows then the attacker now has probable cause to believe the target is running Windows.

Figure 63 represents the histogram for the combined test results for Nmap and Nessus against each host per trial. This histogram shows that thorough obfuscation must obfuscate all layers of a network packet. OSfuscate only focuses on the TCP and IP header portions of the packet and does not obfuscate Application layer protocols or other transport layer protocols making operating system identification possible. The SNOS program defeats Nmap's and Nessus' fingerprinting techniques by obfuscating all layers of a network packet. Figure 63 shows that no variance occurrs within each trial, the

97

benchmark and OSfuscate trials always identify the Windows operating system and the

SNOS trial never identifies the Windows operating system.
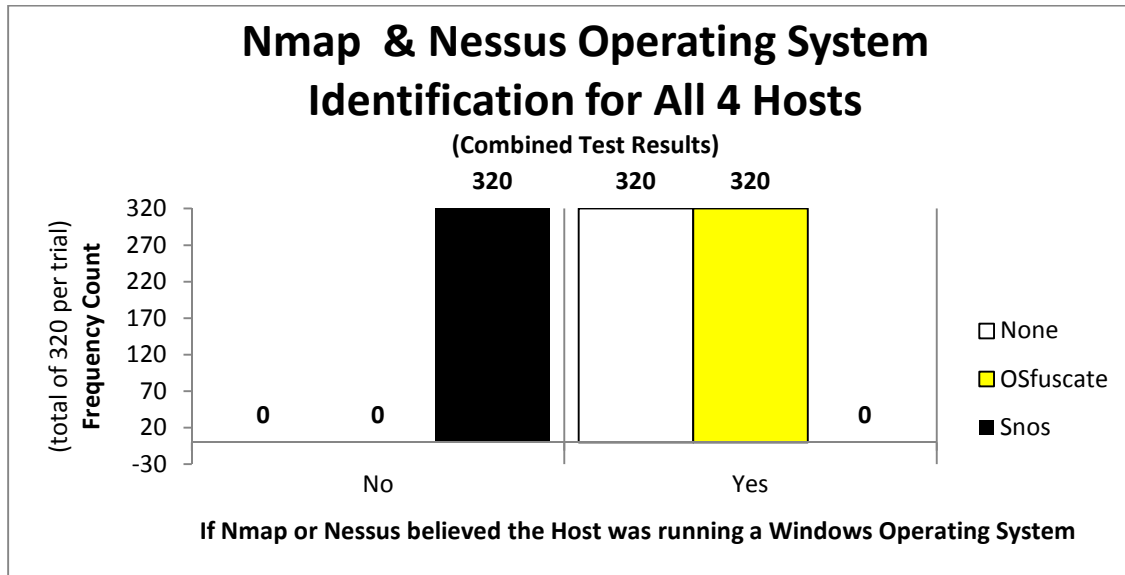
Figure 63: Combined Test Results for Nmap and Nessus

5.4.1   Nmap Analysis

After completing forty repetitions, Nmap produced 320 test results, counting the

two Nmap tests individually, for each of the three trials – benchmark, OSfuscate, and

SNOS and all four hosts. Figure 64 shows the histogram separated by each trial for the

combined test result. The OS Class and the Service tests are combined to a single result,

making 160 results per trial for Nmap. Combined test results are calculated such that if

one or both of the tests result in a successful operating system identification than the

combined result is considered successful.

Appendix D shows all the results for each Nmap test and the combined result of

the two tests for each run. If a Y exists in either the OS Class or Service columns then

the Combined column contains a Y. The Combined column highlights the importance of

a thorough network packet obfuscation process. The Combined column from Appendix D is represented in Figure 64.

The combined Nmap test result correctly identifies the host 100% of the time for both the benchmark and OSfuscate trials and 0% of the time for the SNOS trial. Figure 64 shows that the SNOS program is 100% successful against Nmap fingerprinting because Nmap is never able to identify the Windows operating system. The failure of Nmap to identify the correct operating system is a result of the SNOS program running during the SNOS trial because the SNOS program is the only variable difference between the benchmark and SNOS trials. The OSfuscate program produces the same results as the benchmark; therefore, OSfuscate providesw no additional obfuscation benefits over the benchmark trial.
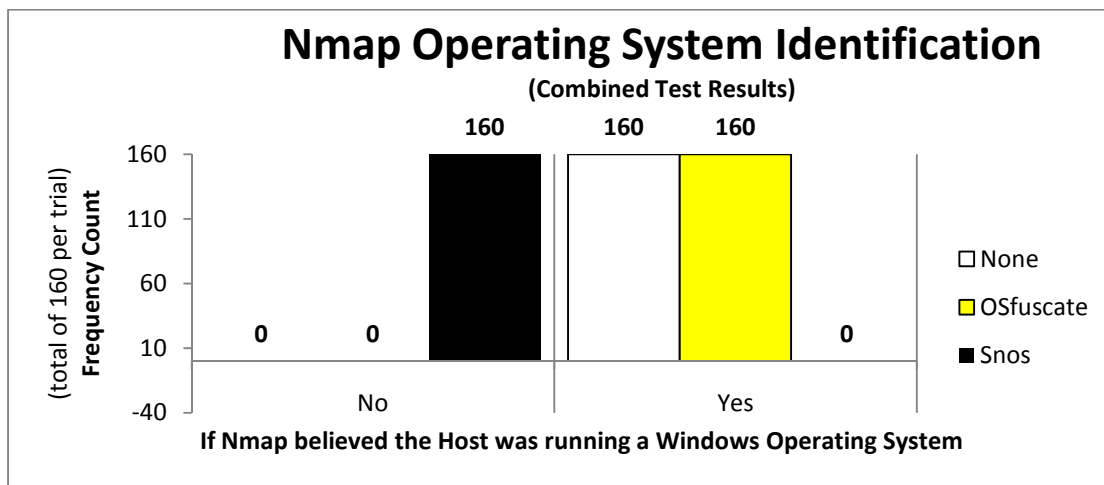
Figure 64: Combined Test Results for Nmap Operating System Identification

Figure 64 results are significant because both tests must be defeated to completely obfuscate a host's operating system against Nmap. Analyzing the results by test – OS Class and Service – for the Workstation host provided additional results worth noting, even though the combined test result provides the important obfuscation effectiveness

99

result. The Nmap Service test on the workstation host is the only time out of the 960

Nmap Service results that variation occurred within a trial's results.

Figure 65 shows how both the benchmark and OSfuscate trials produce "No"

results because the Nmap Service test is not able to detect the SMB (port 445) version

running on the host. The Nmap Service test identifies the SMB version on every run in

the benchmark and OSfuscate trials for the remaining three hosts – Exchange server,

SharePoint server, and web server. SMB is the only service running on the workstation

that would identify the underlying Windows operating system using the Nmap Service

test. (The SSH service running on the workstation provides no additional information

about the underlying operating system because the exact same version of SSH server

running on the workstation, natively runs in a Linux environment.)

Fifteen results from the benchmark trial on the workstation host return no version

information about the SMB protocol (and therefore, no information about the underlying

operating system). The OSfuscate trial yields fourteen results in which the SMB version

is not identified by the Nmap Service test. SNOS is able to obfuscate the SMB version

throughout all forty experimental runs for the workstation as seen in Figure 65.

Possible reasons that the Nmap Service might not occasionally identify the SMB

version during the benchmark and OSfuscate trials is related to network latency – lost or

corrupted network packets. The necessary network packets for the Nmap Service to

identify the SMB version do not arrive in either a timely manner or a manner consistent

with what Nmap is expecting. The Nmap Service test uses a timing delay to identify

services based on the response from custom packets sent to a host. Packets received past

the time allowed are no longer considered part of the fingerprinting process. The timing

delay used for the Nmap Service test expires before the response is received back from the workstation host.
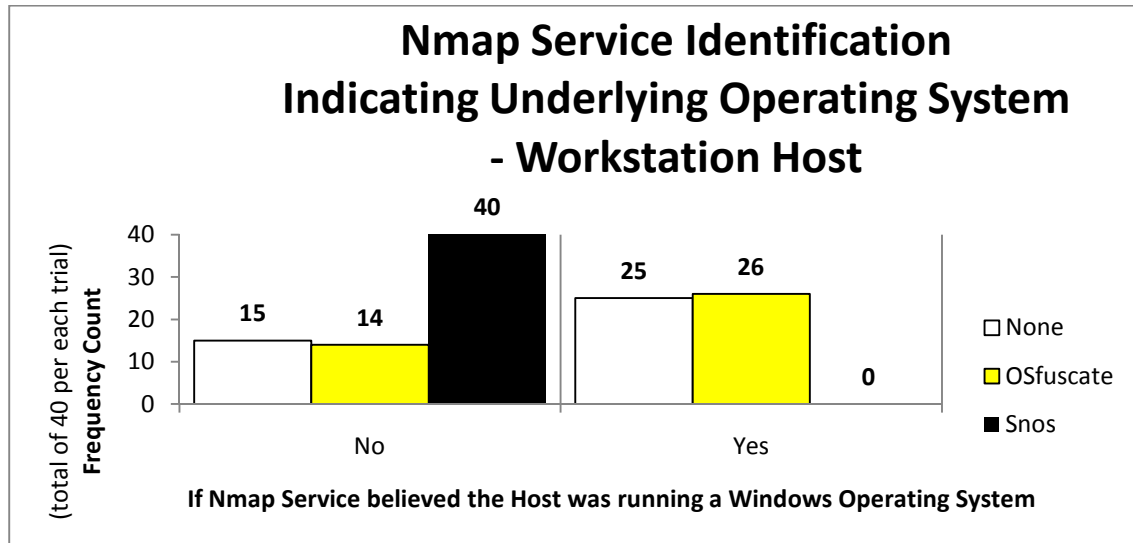
Figure 65: Workstation Host Result from Nmap Service Test

The variance in the results shown in Figure 65 is not significant because the combined test results provide the significant obfuscation analysis. A successful deception process, Section 2.2.1, must be verifiable from multiple sources, meaning that each test for Nmap and Nessus must provide effective obfuscation results. The combined test results for the workstation host are always successful at identifying the Windows operating system.

Using the 2x2 contingency table in Table 6, Fisher's Exact Test (FET) is run from the results shown in Figure 65. The null hypothesis was that OSfuscate is the same as the benchmark trial. Conversely, the alternate hypothesis is that OSfuscate does not equal the benchmark. FET is calculated by using the data in Table 6 with the formula following formula: $[(R_1!R_2! \cdots R_m!)(C_1!C_2! \cdots C_n!)] / [N! \prod_{ij} a_{ij}!]$ where R variables represent the row totals, C variables represent the Column totals, N variable is the total

number of results – the total/total cell in Table 6.   The Fisher's Exact Test returns

0.17885 meaning that with a 95% confidence interval, the null hypothesis cannot be

rejected because 0.17885 > 0.05. As expected, the results between the benchmark and

OSfuscate trials cannot be determined to be different for the workstation from the Nmap

Service test. The benchmark trial actually produces more results that do not identify the

Windows operating system than the OSfuscate trial. The FET calculation means that no

significant difference is detected between the benchmark and the OSfuscate trials despite

the occasional failure of the Nmap Service to identify the SMB version.

Table 6: 2x2 Contingency Table – Benchmark and OSfuscate - Workstation

	Benchmark	OSfuscate	Total
Not Identified	15	14	29
Identified	25	26	51
Total	40	40	80

The Nmap OS Class test's confidence level is shown in Figure 66. Nmap's

confidence level represents how confident Nmap is in matching the results to a known

operating system fingerprint in Nmap's database. Figure 66 represents the results for all

fours hosts combined (forty results per host) and shows that OSfuscate does affect

Nmap's confidence level. A specific confidence level range is not considered significant

because additional research would need to determine at which confidence level

percentage an average user no longer trusts the result. If the host's operating system is

102

correctly identified by the test, no matter the confidence level, the test is considered successful in identifying the Windows operating system.

Ultimately the Nmap OS class test, running OSfuscate, identifies the host operating system 100% of the time. (Windows is almost always identified with the highest confidence level and always greater than an 88% confidence level). Nmap has a 0% confidence level in identify Windows during the SNOS trial because Nmap OS Class never identifies the Windows operating system.
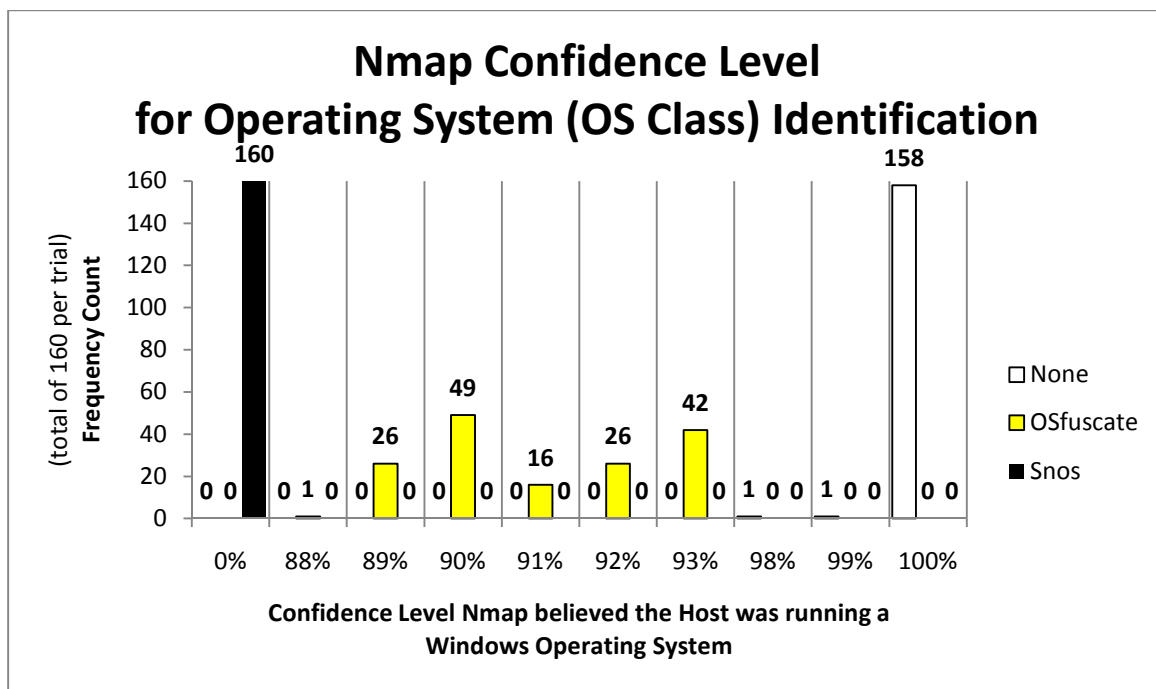
Figure 66: Nmap OS Class Confidence Level of Accuracy for Windows Operating System Fingerprinting

Appendix E contains the list of histograms for the combined Nmap tests' results for each host – Exchange, SharePoint, web, and workstation. The histograms show a visual difference between the SNOS trial's and the other two trials' results. The only difference between each trial is the obfuscation program running on the host – the benchmark trial has no obfuscation program, the OSfuscate trial has the OSfuscate

program running, and the SNOS trial has the SNOS program running; therefore, any statistically significant difference between the results can be attributed to the obfuscation program running.

Table **7** is the 2x2 Contingency Table for the Combined test results between the benchmark and SNOS trials and shows the data used to calculate Fisher's Exact Test for each host individually per fingerprinting program. Each fingerprinting program produces identical combined test results shown in Appendices D and F.

Table **7** represents the 2x2 Contingency Table for either Nmap's and Nessus' results because the combined test results are identical and represents all four hosts because the combined test results for each host are identical. Using the Fisher's Exact Test formula, the resulting p-value is 2.75168E-21 – an extremely small number. This p-value is significant which means the null hypothesis – that the trials are equal – can be rejected and the alternate hypothesis that the trials are different can be accepted.

The FET calculation shows the statistical significance between the results of the SNOS trial and the benchmark trial. This calculation confirms that the SNOS program causes the difference in the results for each of the four hosts. This experiment emulates a production environment but is not a randomly chosen configuration setup out of all possible configurations for each service and host. The scope of interference is limited to the configurations used during this experiment.

Table 7: 2x2 Contingency Table – Benchmark & SNOS – Combined Test Results

	None	SNOS	Total
Not Detected	0	40	40
Detected	40	0	40
Total	40	40	80

5.4.2   Nessus Analysis

The combined test results for both Nessus tests – OS Class and Service – are shown in Appendix F. Nessus yields 960 results, 320 per trial, between both the OS Class and Service tests. The only recorded difference between the results of Nessus and Nmap is that the Nessus Service test is able to correctly identify the version of all tested services on each host every time, and subsequently identify the underlying operating system 100% of the time, for the baseline and OSfuscate trials. The Nmap Service test occasionally fails to identify the service version for the baseline and OSfuscate trials as previously discussed in Section 5.4.2 for the workstation host.

Figure 67 is a histogram representing the combined test results for each trial with Nessus. The results within each trial, like Nmap, provide no variance. Nessus correctly fingerprints the Windows operating system during the baseline and OSfuscate trials. Nessus does not fingerprint the Windows operating system during the SNOS trial. The lack of variance within each trial's results provides a clear picture of the effectiveness of each trial in defeating Nessus.   Appendix G contains the histograms of the Nessus combined test results for each host.

Table **7** also represents the 2x2 contingency table for the Nessus combined test results for each host. This 2x2 contingency table produces a p-outcome value by using Fisher's Exact Test algorithm. The p-value is the same as Nmap's combined test's p-value because the results are identical and is 2.75168E-21 which is significantly smaller than the statistically significant p-value of .01. The small p-value means that a statistically significant difference exists between the SNOS and baseline trials. The FET result statistically proves that the SNOS program causes the difference between the two trials because the SNOS program is the only changed variable between each trial.
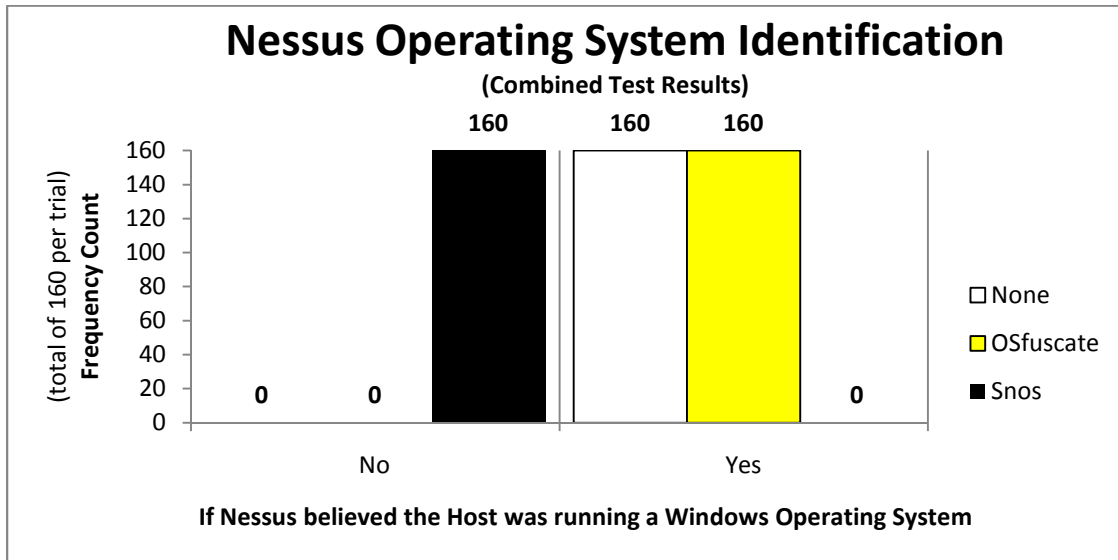
Figure 67: Combined Test Results for Nessus' Operating System Identification per Trial

The previous histograms show that SNOS is effective in defeating Nmap and Nessus. The experiment was repeated forty times. After the initial twenty runs, no variance was detected in the combined test results. An additional twenty runs were performed to increase the combined test results from twenty to forty for each host to provide a larger result set to analyze.

If after forty runs, the variance was still highly volatile, meaning the standard deviation for each trial was not converging on a particular number then the experiment would have been repeated until the variance and standard deviation converged towards a specific number – meaning the variance was no longer volatile. Forty repetitions were sufficient because each trial's variance was not volatile – providing no variance within each trial for the combined test results. Forty repetitions were sufficient to determine the significance of the results of the obfuscation effectiveness experiment as shown by the histograms and proven by the Fisher's Exact Test calculations.

5.4.3   Combined Obfuscation Analysis

OSfuscate does not obfuscate the host operating system during the experiment because OSfuscate is created to only defeat the TCP/IP scanning method of fingerprinting by changing Windows registry values. Both Nmap and Nessus use several techniques to fingerprint a host's operating system – more than just TCP/IP fingerprinting.

As discussed in Sections 5.4.1 and 5.4.2, the results from the obfuscation effectiveness experiment for the SNOS trial always obfuscates the Windows operating system. SNOS achieves a 100% obfuscation success rate is because of the finite set of methods used to identify the host's operating system. Considerable time was spent identifying and confirming the various fingerprinting methods. Once these methods and differences were identified, SNOS was written to effectively obfuscate the protocols to counter the techniques used by Nmap and Nessus. SNOS is developed to allow future additional protocols to be obfuscated by SNOS as needed.

107

5.5   Network Latency Analysis

The network latency experiment consists of two trials – the baseline and SNOS. The network latency experiment is repeated ten times and yields 960 individual results – 480 results for the baseline trial and 480 results for the SNOS trial. Each host has 120 individual results per trial. The analysis of the network latency experiment is divided by host in the following subsections. The generated network load for each trial over the course of each experimental run is identical by using Tcpreplay. The network latency is recorded in microseconds.

The sample network traffic represents specific traffic for each host from an entire day's network load. The network traffic originates from the Cyber Defense Exercise (CDX) 2010 traffic at the Air Force Institute of Technology. Obtaining a random sample from all possible network loads is not feasible; therefore, the results and analysis do not apply to all possible network loads. Even if a random sample is taken from the samples available, inference to the general population of all possible configurations could still not be made. The complete samples available do not represent a random sample from all possible hosts and configurations currently employed throughout the world for the selected protocols and services. The following analysis is constrained with these conditions.

Virtualization and Tcpreplay allow each trial and each repetition to be identical. Causalities can be inferred regarding the effect the SNOS program has on network latency for each host in reference to the scope of inference noted.

5.5.1 Exchange Host Analysis

Figure 68 shows all 120 individual results for the Exchange host during the baseline trial, and Figure 69 shows the SNOS trial results. The dots in each figure represent the individual results for each trial. The black lines show the mean from all 120 results for each trial.
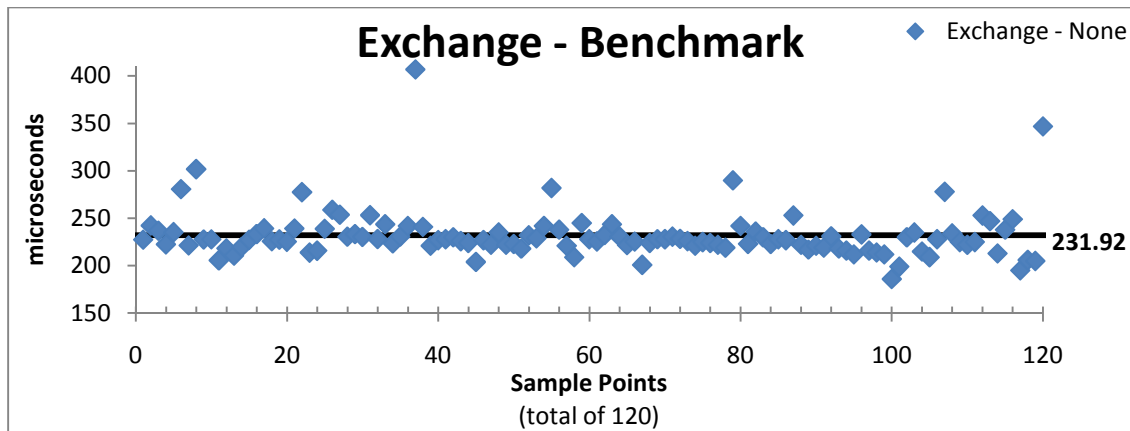
Figure 68: Network Latency – Exchange – Benchmark
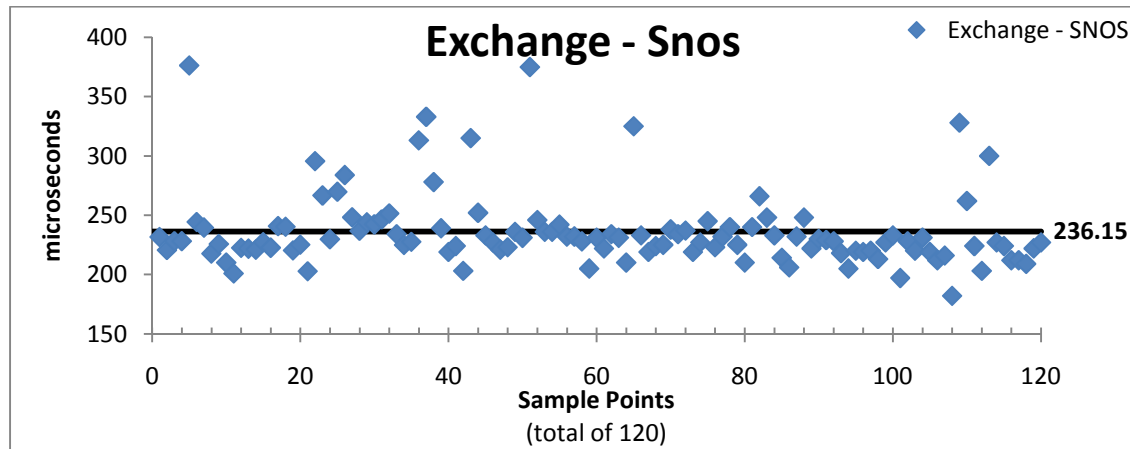
Figure 69: Network Latency – Exchange - SNOS

Figure 70 shows the confidence intervals and means as a comparison of means. The mean for the benchmark trial is 231.92 microseconds and the mean for the SNOS trial is 236.15 microseconds. The benchmark has a tighter 95% confidence interval with

only a +/- 4.7 microseconds range from the mean, Figure 70. The SNOS trial's 95%

confidence interval is 5.66 microseconds +/- range from the mean. The confidence

interval, +/- 4.7 or +/- 5.6, is added to each respective mean to graph the rectangles in

Figure 70. The number outside the upper corner of the rectangle represents the upper

range of the confidence interval and the number outside the lower corner represents the

lower range of the confidence interval.

The mean for the benchmark trial is barely within the 95% confidence interval

range for the SNOS trial and vice versa. Since the means fall within each other's

confidence intervals, the trials' results cannot be considered different.
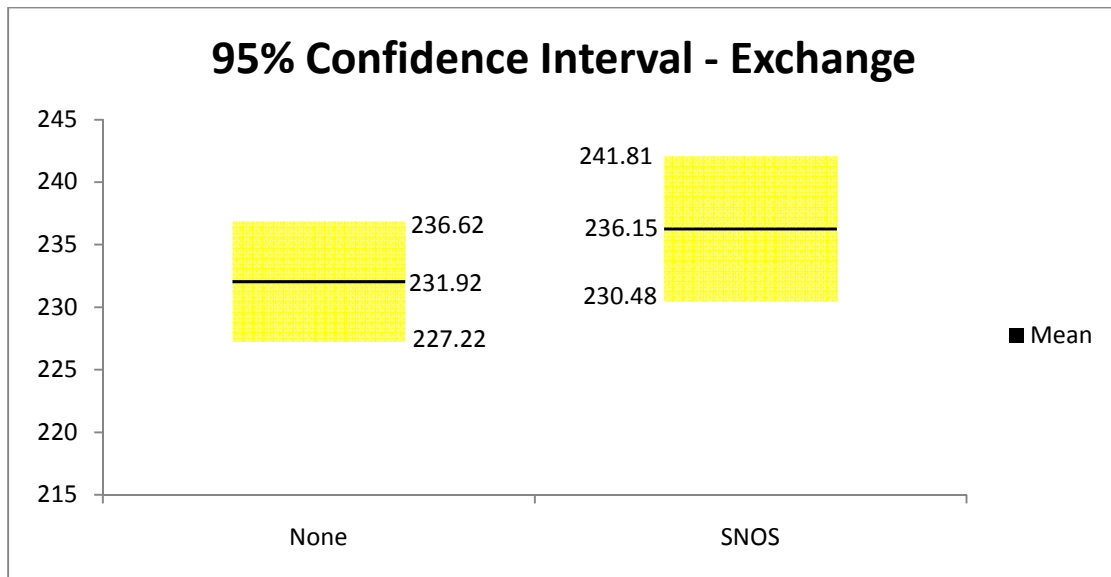
Figure 70: 95% Confidence Interval – Exchange

The t-Test calculation, Table 8, confirms the Comparison of Means that at the

95% confidence level no significant difference exists between each trial's results.

110

Table 8: Exchange Server t-Test

	Benchmark	SNOS
	227.4066667	*231.6533333*
Mean	231.9620168	236.1862465
Variance	695.4812955	1010.461557
Observations	120	120
Pearson Correlation	0.243925237	
Hypothesized Mean Difference	0	
df	119	
t Stat	-1.279544022	
P(T<=t) one-tail	0.101607858	
t Critical one-tail	1.657869523	
P(T<=t) two-tail	0.203215716	
t Critical two-tail	1.980272226	

The box plot, shown in Figure 71, shows the variation of the results within each trial and which trial had a larger spread. The stars indicate the largest maximum outlier, if any, in the results. The triangles represent the smallest minimum outlier. In the box plot, only the maximum and minimum outliers are shown. 50% of the results fall within the box for each trial. The middle line is the median for each trial and means that 50% of all results for a trial fall above and 50% fall below the line.

Table 9 is the Exchange data that is used to create Figure 71 and shows the number of outliers for each. The IQR (interquartile range) row in Table 9 shows that SNOS has more variance, 19.5, than the benchmark trial, 14.5. The spread is shown graphically in Figure 71 by the thickness of each box plot; SNOS has a thicker box plot.
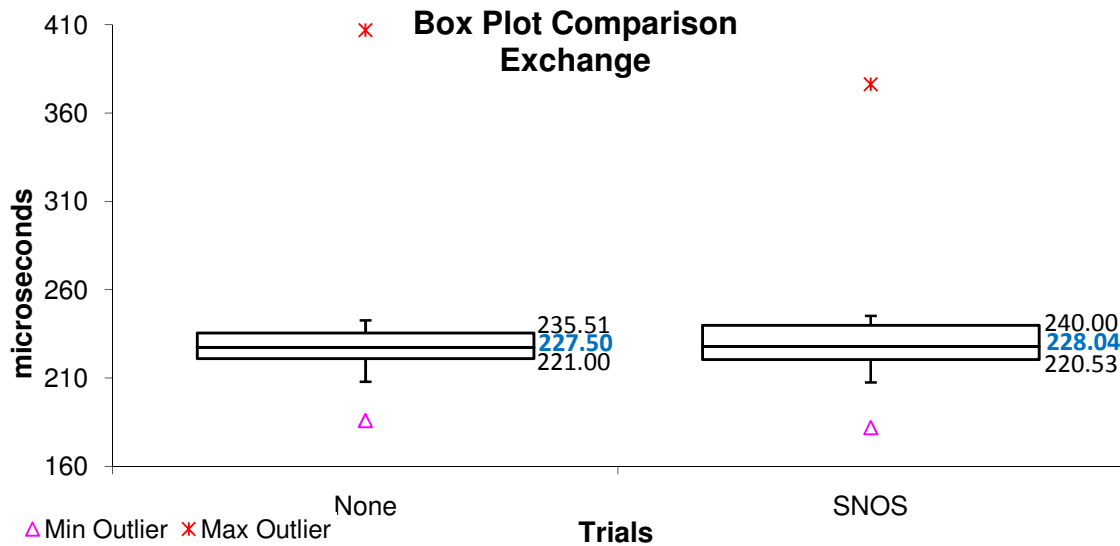
Figure 71: Box Plot of Exchange Results

Table 9: Box Plot Data - Exchange

	None	SNOS
Min	186.00	182.00
Q_1	221	220.5333333
Median	227.4966667	228.035
Q_3	235.51	240
Max	407	376.2933333
IQR	14.51	19.46666667
Upper Outliers	9	12
Lower Outliers	3	1

Table 10 is the ANOVA, Analysis of Variance, between the benchmark and SNOS trials. The ANOVA is an analysis of the variance by comparing the means. The analysis uses a 95% confidence interval. The null hypothesis is that no difference exists between the benchmark and SNOS trials. The summary section, in Table 10, is a summary of the results per trial: "Count" is the number of results, "Sum" is the summation of all the results per trial, "Average" is the mean, and "Variance" is the variance per trial.

The ANOVA portion of Table 10 shows the statistical information from the F-test and ANOVA. The df represents the degrees of freedom, which is one less than the number of groups, 2 – 1. The SS column represents the sum of squares between groups and within groups. The F crit value was calculated using the degrees of freedom. The F value is calculated by squaring the t-statistic. (The t statistic is squared because the F-test is a one sided test). Microsoft Excel calculated the exact F value. The F value can also be looked up in a t value table. Using an F-Distribution lookup table with $\alpha = 0.05$, the F crit value = 3.88 by looking in the table when v1 = 1 (degrees of freedom between the groups) and v2 = 238 (degrees of freedom within the groups). The exact F crit value calculated by Excel is shown in Table 10. α is selected to be 0.05 meaning that the results are significant at a 5% significance level.

The null hypothesis cannot be rejected because F crit is greater than F value in Table 10. The p-value represents how likely the difference between the two means occurrs by chance. The lower the p-value the less likely the difference is a result of chance. The large p-value, 0.26171, means that the results could have occurred by chance. A p-value less than the significance level is interpreted to mean the difference is unlikely to have occurred by chance. The ANOVA result implies the two trials' means cannot not be considered equal because the null hypothesis cannot be rejected. The ANOVA analysis indicates that the trials' means can be equal but also that the results could have occurred by chance since the p-value was not significantly low. Further analysis is discussed in Section 5.5.5 about the combined network latency analysis for the results of all the hosts together.

Table 10: Exchange ANOVA and F-Test Analysis

SUMMARY

Groups	Count	Sum	Average	Variance
None	120	27830.88667	231.9240556	689.8098417
SNOS	120	28337.81667	236.1484722	1002.141511

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	1070.741771	1	1070.741771	1.26568	0.26171	3.88082
Within Groups	201342.211	238	845.9756763			
Total	202412.9527	239				

5.5.2   *SharePoint Host Analysis*

The results for each trial are shown in Figure 72 and 73, respectively. Each represents a result. Several outliers in the benchmark trial are extreme – over 1000 microseconds. These outliers dramatically influence the analytical results for the benchmark trial. These extreme outliers for the benchmark trial shift the mean and confidence interval. These benchmark outliers are shown in Figure 72 by the dots that took over 1000 microseconds to return.
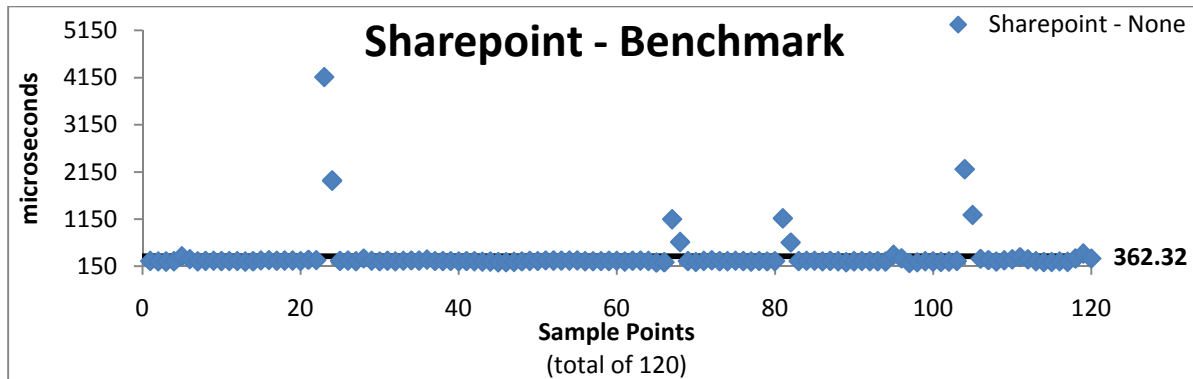
Figure 72:   Network Latency – SharePoint - Benchmark
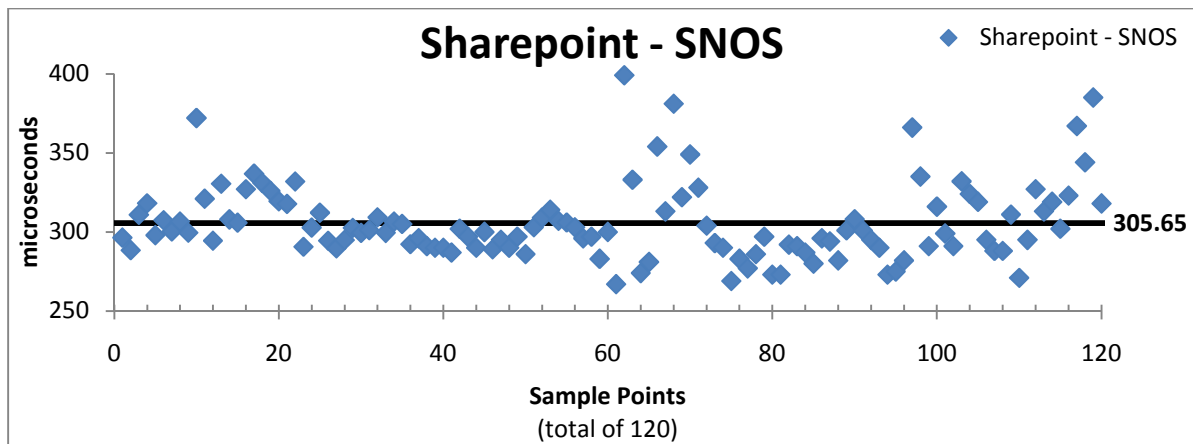
Figure 73: Network Latency – SharePoint - SNOS

The mean for the benchmark trial is 362.32. The mean for the SNOS trial is
305.65. The SharePoint host's results yield a better mean time with the SNOS program
running than the benchmark without the SNOS program. The SNOS trial inspects each
incoming and outgoing packet by an additional program. An additional step to receive
and send network traffic should produce a higher mean time not a lower one. Figure 74
shows a large confidence interval range for the benchmark trial compared to the SNOS
trial for the SharePoint server. Using the comparison of means, the benchmark's mean
does not fall within the SNOS trial's confidence interval.
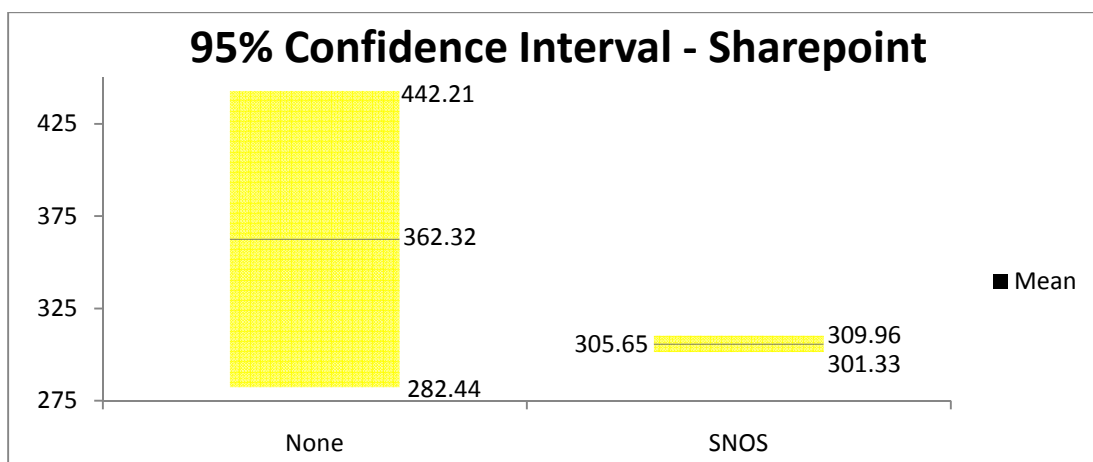
Figure 74: 95% Confidence Interval – SharePoint

115

The extreme outliers in the benchmark trial drastically affect the analysis of the benchmark results by skewing the data, as shown with the large confidence interval in Figure 74. By removing the outliers, the results produce statistically significant differences between the two trials, shown in Figure 75. Without the outliers, the benchmark trial is significantly different than the SNOS trial because neither mean falls within the other's confidence interval. This comparison of means provides statistical evidence for a difference between the two trials in network latency when the extreme outliers are removed.

Removing the outliers improves the statistical analysis of the results but can possibly throw out valid data. If the outliers that are removed are a result of network problems not related to the actual trial then the removal can be warranted. For instance, if a NIC is experiencing problems and results in the outliers than removing the outliers could make statistical sense without damaging the integrity of the results. This scenario is unlikely because extreme outliers happen over the course of several repetitions during the benchmark trial and never during the SNOS trial. The outliers could have resulted because of normal network latency based on the specific packets requested from the SharePoint server. These packets might not have affected the SNOS trial because the SNOS program could have altered the packets such that the operating system's performance was enhanced.

SNOS is programmed to drop specific types of incoming request packets. A large number of these packets would require a large number of responses from the SharePoint server. The benchmark trial would create these responses while the SNOS trial would

116

drop the request packet and the Windows operating system and underlying application would not create a response packet.

The extreme outliers during the benchmark trial happened when the SharePoint server received several SMB Null Session requests and subsequent SMB queries trying to use the Null Session connection. Several of these SMB queries were queries for objects that did not exist on the SharePoint server. During the benchmark trial, the SharePoint server spent time processing and executing these SMB queries which accounted for the extreme outliers. These Null Session requests attempted to access the sysvol and IPC$ default shared directories and attempted to query inside Window policy files and objects. Most of the policy objects and files did not exist within the SharePoint server meaning that during the benchmark trial these requests took additional time as Windows queried and searched for these objects. These incoming requests come from the traffic taken from the Cyber Defense Exercise of 2010 and were used to try and gain access into the hosts during that exercise. SNOS effectively dropped the original SMB Null Session request and queries which eliminated the Windows processing time searching for the invalid objects queried.

Looking at Figure 72, the extreme outliers are significantly above the otherwise almost flat line response from the remaining results. Four out of ten repetitions produce extreme outliers.
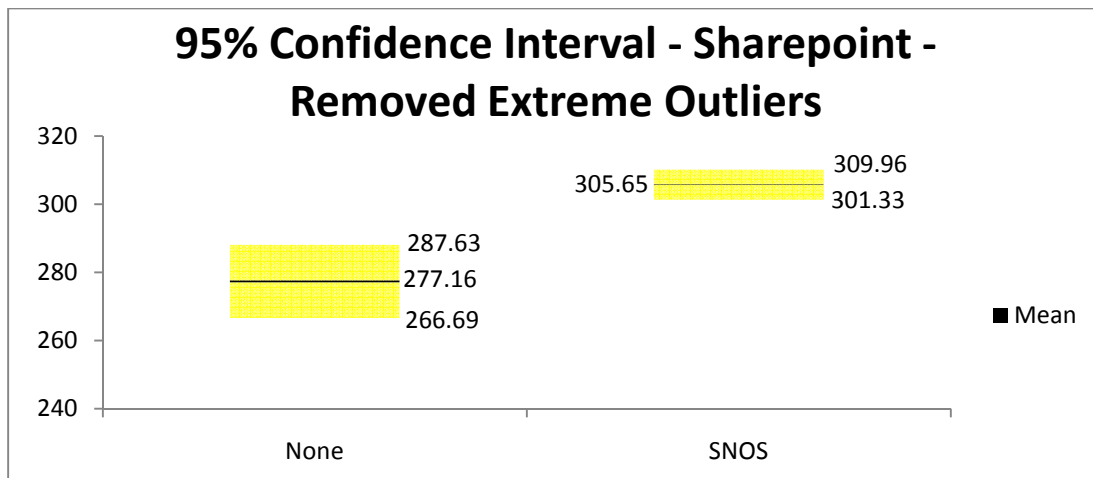
Figure 75: 95% Confidence Interval – SharePoint without Outliers

Ignoring the validity of the outliers, Figure 75 provides conclusive evidence that a

difference exists between the network latency of the two trials; however, as previously

stated, the extreme outliers during the benchmark trial are a result of SBM Null Session

packets that queried invalid objects on the SharePoint server.

The box plot, Figure 76, shows a comparison between the spread of the two trials'

results. Table 11 shows the data used for the box plot and the number of maximum and

minimum outliers for each trial. The box plot includes all results, even extreme outliers,

for each trial. The IQR (interquartile range) value in Table 11 shows that the SNOS trial

has more variance, 25, than the benchmark trial, 19. The benchmark has less spread even

though the benchmark trial has several extreme outliers that skewed the mean and

confidence intervals. The box plot and its data provide an unbiased view, irrespective of

outliers, about the results between each trial. This data shows that most of the time, on an

individual result basis, SNOS increases the network latency. Over 75% of SNOS' results

have more network latency than 75% of the benchmark results. Figure 76 does not show

the maximum outlier for the benchmark trial because the maximum outlier, 4164.58, lies

off the graph and the graph is narrowed down to more clearly depict the difference

between the box plots of the two trials.
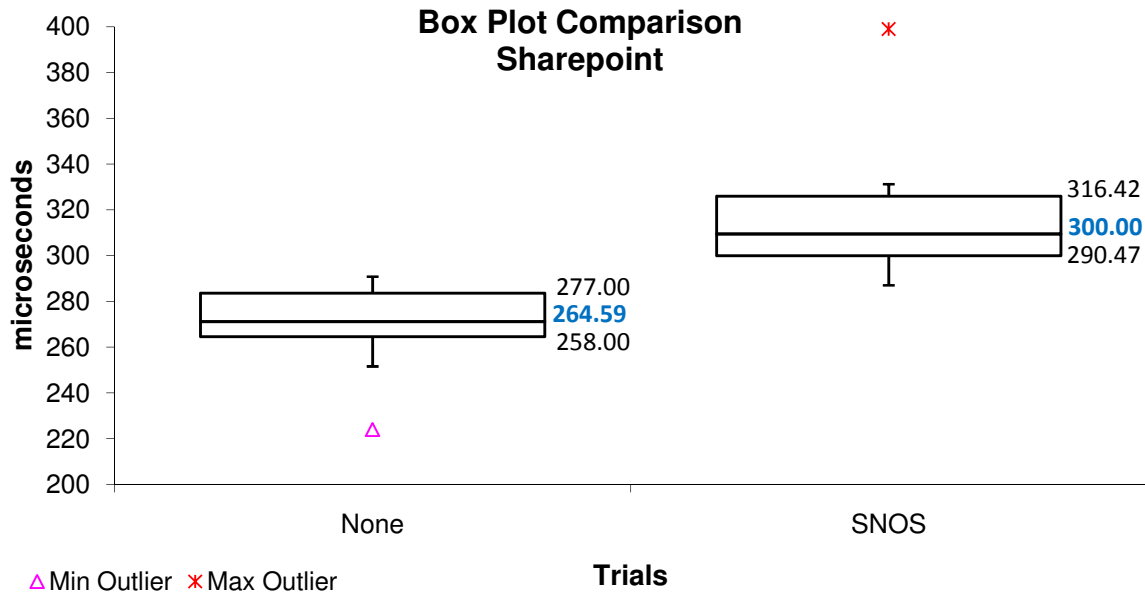
Figure 76: Box Plot for SharePoint Results

Table 11: Box Plot Data - SharePoint

	None	SNOS
Min	224.00	267.00
Q_1	258	290.47
Median	264.59	300
Q_3	277	316.4241667
Max	4164.58	399
IQR	19	25.95416667
Upper Outliers	17	6
Lower Outliers	1	0

Table 12 shows the ANOVA table for the SharePoint results which analyzes the

significance of the results. The null hypothesis is that no difference in network latency

exists between the benchmark and SNOS trials. The results of the F-Distribution values,

F and F crit, mean that the null hypothesis cannot be rejected – the benchmark and SNOS

trials cannot not be considered equal. The significance level for the SharePoint ANOVA

test is 0.05, α = 0.05. The p-value of 0.16627 is high which means the difference could

have occurred by chance and could not be determined to be statistically significant. The

inconclusive ANOVA result is because of the extremely high outliers during the

benchmark trial. The benchmark trial yields six results that are over 1000 microseconds

and about three times larger than the median and 75% of all the remaining results. These

five results create a long-tailed distribution which inherently causes problems with

ANOVA.

Table 12: SharePoint ANOVA and F-Test Analysis

SUMMARY

Groups	Count	Sum	Average	Variance
None	120	43478.59	362.3215833	199347.2767
SNOS	120	36677.47667	305.6456389	582.0027408

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	192729.7607	1	192729.7607	1.92797	0.16627	3.88082
Within Groups	23791584.25	238	99964.6397			
Total	23984314.01	239				

5.5.3 Web Host Analysis

The web server host produces 120 results per trial. The results for the benchmark

and SNOS trials are shown in Figure 77 and 78, respectively. The SNOS trial produces

more extreme outliers than the benchmark trial. The SNOS outliers are not significant to

skew the data similar to the SharePoint benchmark results.
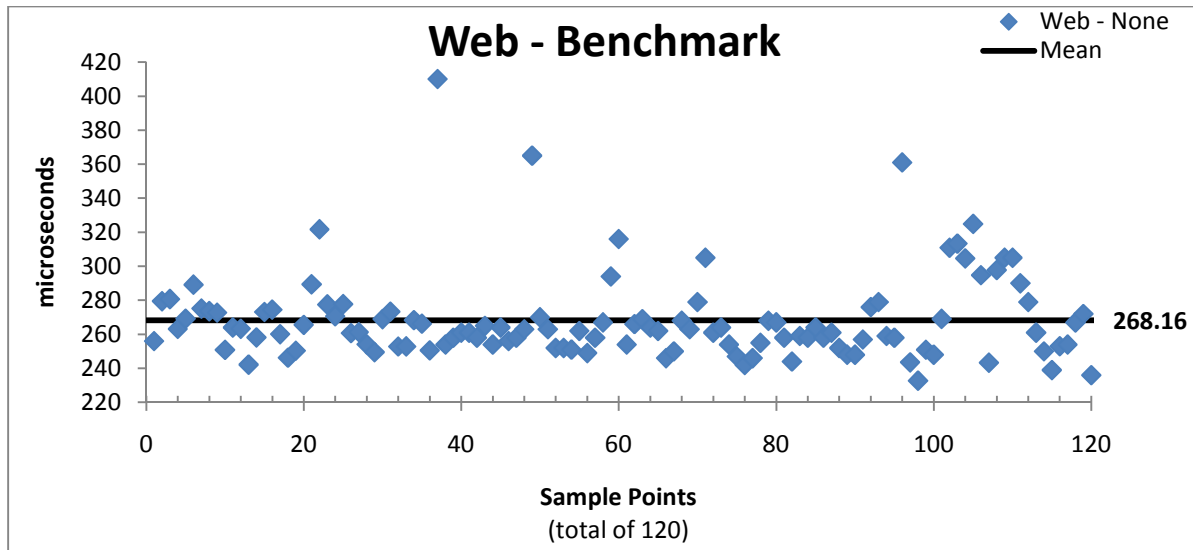
Figure 77: Network Latency – Web - Benchmark
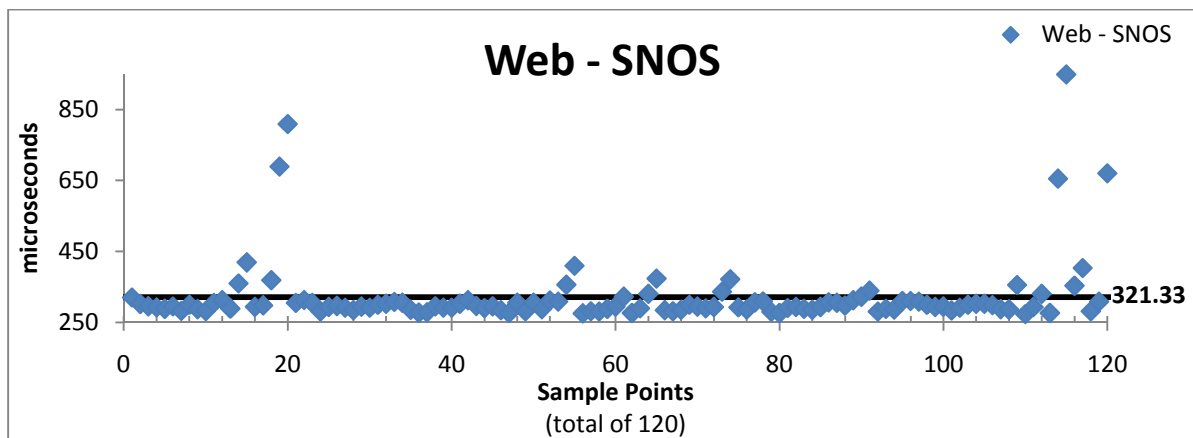
Figure 78: Network Latency – Web - SNOS

The mean for the benchmark trial is 268.16 and 321.33 for the SNOS trial. The

95% confidence interval for the benchmark does not reach the lower range of the 95%

confidence interval for the SNOS trial as shown in Figure 79. The comparison of means

between the two trials is significant such that the SNOS trial can be attributed to being

different than the benchmark trial. Since the network latency difference is significant

because the means do not overlap with each other's confidence interval, the SNOS

121

program can be said to have caused the network latency because the only difference

between the benchmark trial and the SNOS trial is that the SNOS trial has the SNOS

program running.
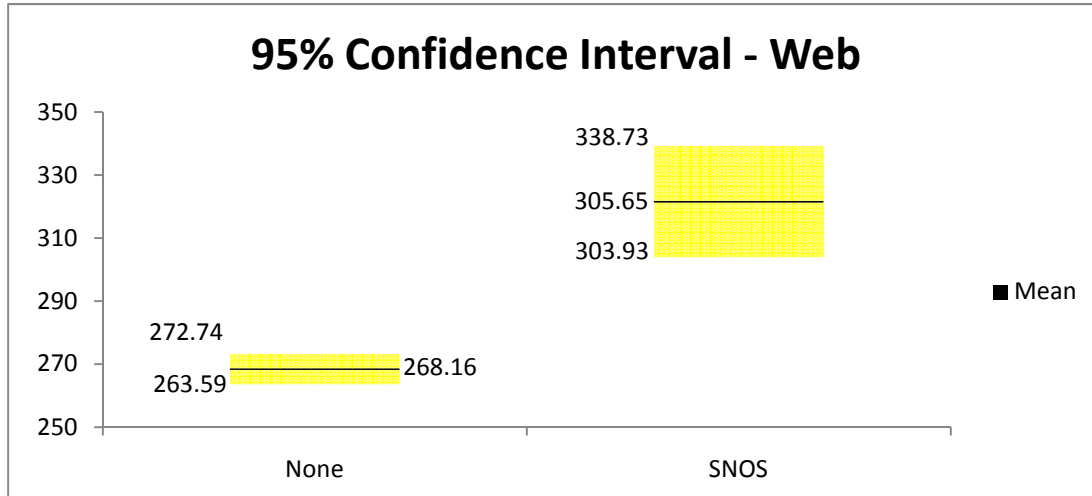
Figure 79: 95% Confidence Interval – Web

The t-Test for the web server, Table 13, further shows the statistical significance

between the differences of each trial's results. The p-value is extremely small and shows

significance even at a 99% confidence level.

Table 13: Web Server t-Test

	Benchmark	SNOS
	256.0533333	*319.79*
Mean	268.2651261	321.3422969
Variance	659.012835	9540.194619
Observations	120	120
Pearson Correlation	-0.183323888	
Hypothesized Mean Difference	0	
df	119	
t Stat	-5.491073838	
P(T<=t) one-tail	1.16381E-07	
t Critical one-tail	1.657869523	
P(T<=t) two-tail	2.32762E-07	
t Critical two-tail	1.980272226	

The box plot in Figure 80 shows the spread of the results within each trial. The median value, 262 microseconds, for the benchmark trial is smaller than the SNOS median, 295 microseconds, meaning that 50% of the results from the SNOS trial produce higher latency times compared with the benchmark trial. The third quartile for the benchmark trial in Table 14 means that 75% of all results have less latency time than 75% of the SNOS trial results. This data is used to produce Figure 80 and indicates that the benchmark trial has fewer, 12, maximum outliers than the SNOS trial, 15. Neither trial yields minimum outliers. An outlier is a result outside either the upper whisker or the lower whisker for each box plot. Figure 80 does not show the SNOS maximum outlier, 949 microseconds, as the graph is only showing results up to 450 microseconds in order to more clearly see the difference between the box plots of the two trials.
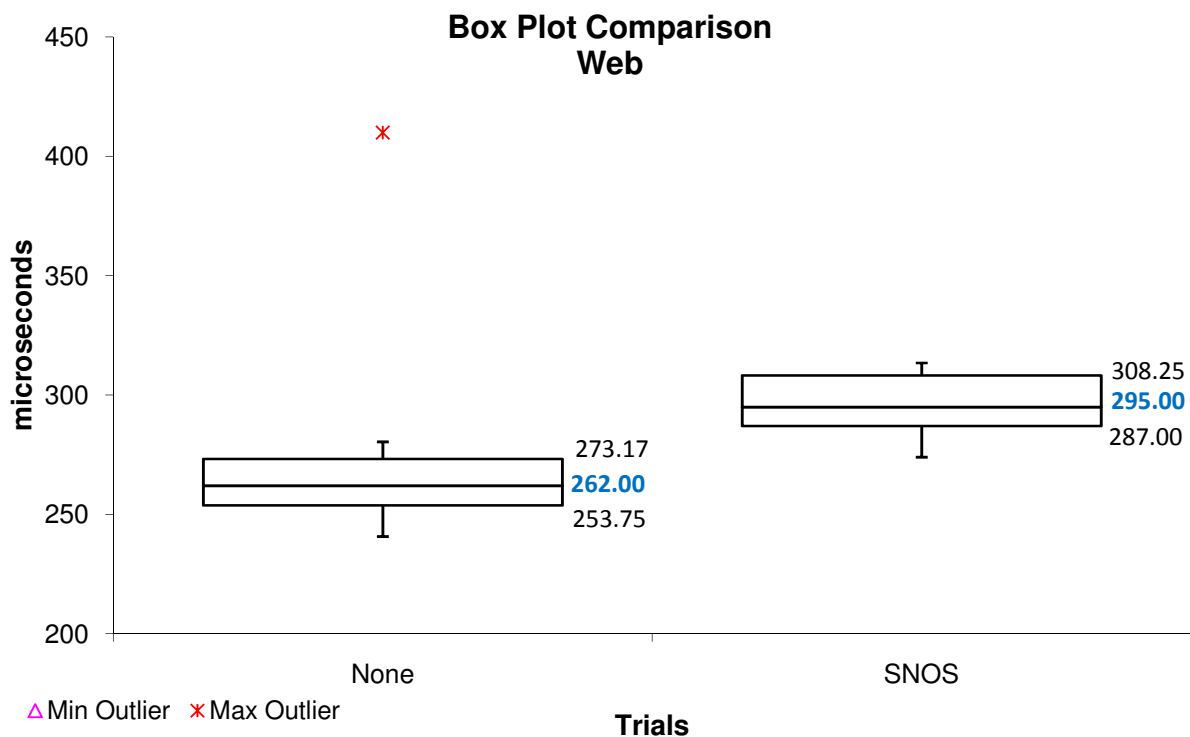
Figure 80: Box Plot for Web Results

Table 14: Box Plot Data - Web

	None	SNOS
Min	232.75	273.00
Q_1	253.75	287
Median	262	295
Q_3	273.1733333	308.25
Max	410	949
IQR	19.42333333	21.25
Upper Outliers	12	15
Lower Outliers	0	0

The ANOVA result, Table 15, shows the statistical significance of the results. The F value is greater than the F-crit value meaning the null hypothesis can be rejected and the alternate hypothesis, that the trials are different, is accepted. The p-value identified is extremely low, 0.00000002998 and indicates that the results are unlikely to have occurred by chance. The ANOVA calculations mean that the SNOS program causes network latency and that the results are highly unlikely to have occurred by chance.

Table 15: Web ANOVA and F-Test Analysis

SUMMARY

Groups	Count	Sum	Average	Variance
256.0533333	119	31924	268.2651261	659.0128
319.79	119	38240	321.3422969	9540.194

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	167622.571	1	167622.571	32.86972	0.000000029980	3.881163
Within Groups	1203506.48	236	5099.603727			
Total	1371129.051	237				

5.5.4 Workstation Host Analysis

Figure 81 and 82 show the results for the benchmark and SNOS trials. Figure 81 shows that the benchmark trial produces more extreme outliers than the SNOS trial. These outliers, unlike the SharePoint benchmark results, are not extreme enough to skew the analysis of the data.
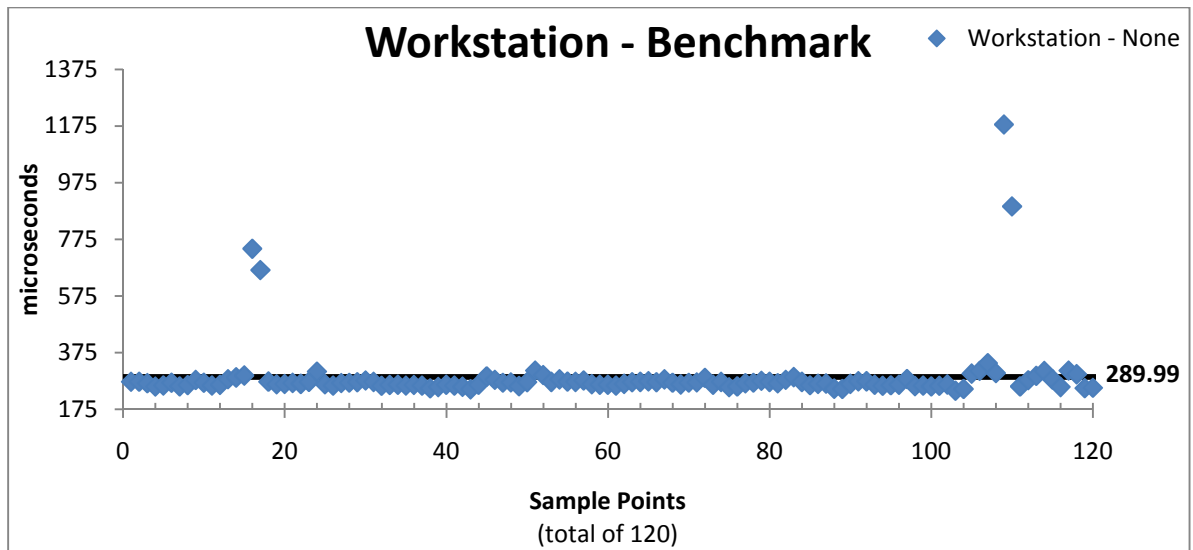
Figure 81: Network Latency – Workstation – Benchmark
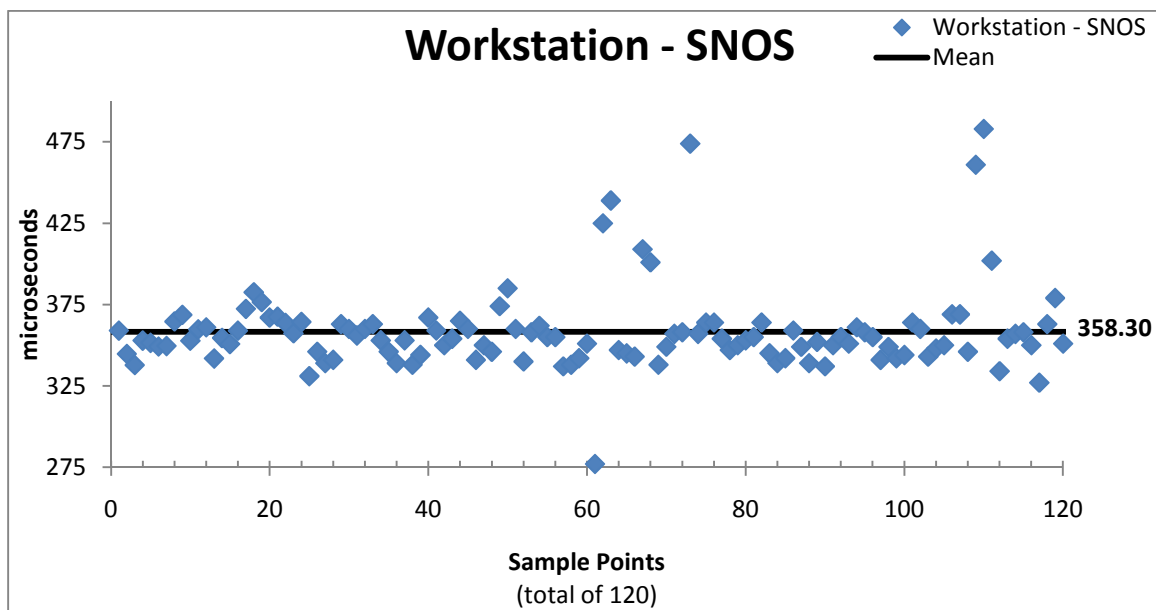
Figure 82: Network Latency – Workstation - SNOS

125

The benchmark trial's mean is 289.99 microseconds which is less than the SNOS's trial mean of 358.3 microseconds. The upper 95% confidence interval line, 269.4 microseconds, for the benchmark trial does not fall within the 95% confidence interval of the SNOS trial. Figure 83 visually shows the comparison of means which provides conclusive evidence that the benchmark trial and SNOS trial results are statistically significant. The SNOS trial produces greater network latency on the network, shown in Figure 83, proven by the comparison of the means. Neither mean falls within the other's 95% confidence interval.
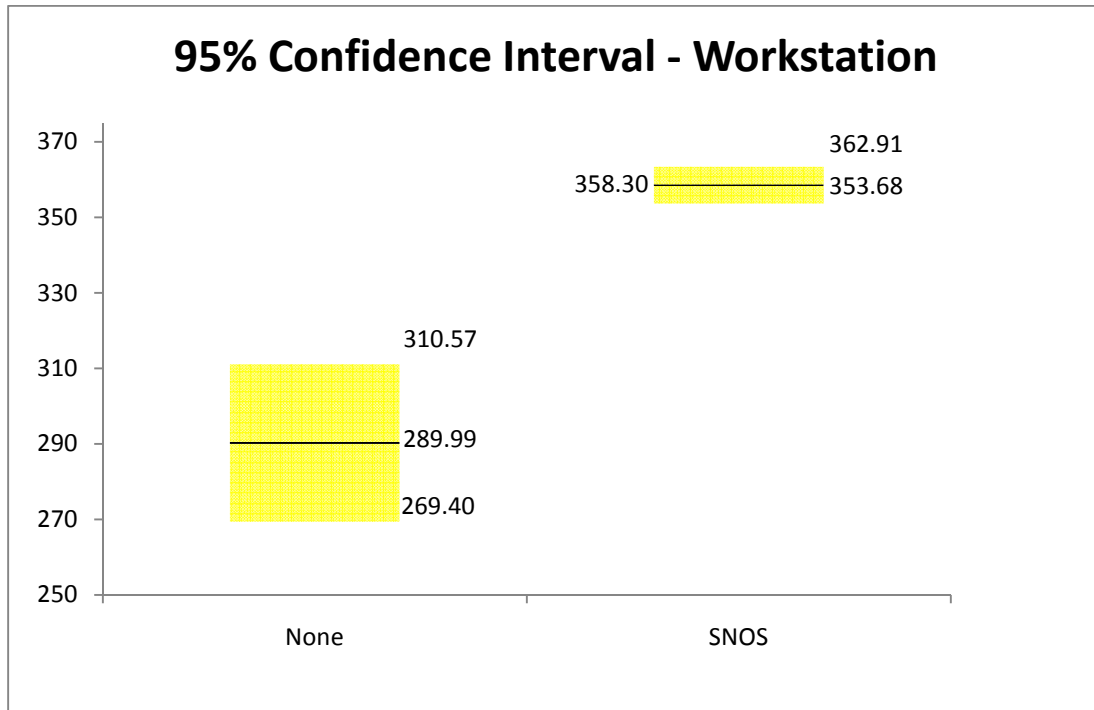
Figure 83: 95% Confidence Interval – Workstation

The Comparison of Means analysis is validated by the t-Test for the workstation host shown in Table 16. The extremely small p-value means that even at a 99% confidence interval the results between each trial would be significantly different.

Table 16: Workstation t-Test

	Benchmark	SNOS
	272.5466667	*358.9833333*
Mean	290.1320448	358.289916
Variance	13344.45857	672.0435035
Observations	120	120
Pearson Correlation	0.502397646	
Hypothesized Mean Difference	0	
df	119	
t Stat	-7.086727531	
P(T<=t) one-tail	5.40025E-11	
t Critical one-tail	1.657869523	
P(T<=t) two-tail	1.08005E-10	
t Critical two-tail	1.980272226	

The box plot, Figure 84, differentiates between the variance of the two trials. The box plot provides a visual difference between the two trials' results and shows that neither the median nor 75% of the data – all results less than or equal to 277 microseconds – are within the interquartile of the SNOS trial. This box plot comparison shows that over 75% of all the results for the benchmark trial has less network latency than 75% of all the SNOS trial results.

Table 17 contains the data used to create Figure 84 and identifies eleven maximum outliers, none of which are shown in Figure 84 because the axis range is narrowed down to more clearly show the box plots. The SNOS trial's minimum outlier, 277 microseconds, is the only result that fell within the interquartile range of the benchmark trial.
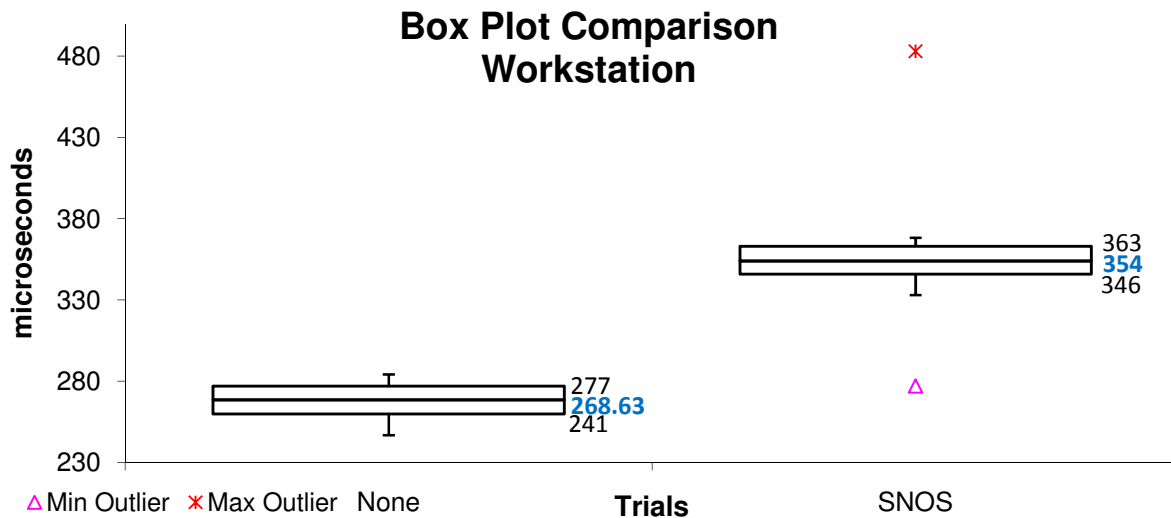
Figure 84: Box Plot for Workstation Results

Table 17: Box Plot Data - Workstation

	None	SNOS
Min	241.00	277.00
Q_1	259.8466667	346
Median	268.6266667	354
Q_3	277	363
Max	1180	483
IQR	17.15333333	17
Upper Outliers	11	8
Lower Outliers	0	1

The ANOVA and F-Test are shown in Table 18. The null hypothesis, that the means between the two trials are equal, can be rejected in favor of the alternate hypothesis that a difference exists between the means. The F value is greater than the F crit value which allows the null hypothesis to be rejected. The p-value, 0.000000001101, is statistically significant. The small p-value means that the results are highly unlikely to have occurred by chance. The F values allow the causation – the SNOS program causes network latency – to be accepted and the p-value means that the results are not by chance.

Table 18: Workstation ANOVA and F-Test Analysis

SUMMARY

Groups	Count	Sum	Average	Variance
None	120	34798	289.9855	13234.89731
SNOS	120	42995	358.2956944	666.400086

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	279976.9599	1	279976.9599	40.28069495	0.000000001101	3.880827
Within Groups	1654254.39	238	6950.648698			
Total	1934231.35	239				

5.5.5 Combined Host Analysis

The four hosts produce varying results regarding the affect the SNOS program has on network latency. The Exchange server results are not necessarily unexpected but simply do not produce conclusive statistical analytical. For the Exchange server SNOS does not cause statistically significant network latency.

Figure 85 and 86 show the combined plots of all four hosts for the two trials – benchmark and SNOS. The first 120 results plotted in each figure represent the 120 results from the Exchange server, the next 120 the SharePoint server, and so on. These figures and further analysis of the combined hosts' results introduce additional variables beyond the statistical relevance of the results. An additional factor is the different network loads that are custom generated according to each host.
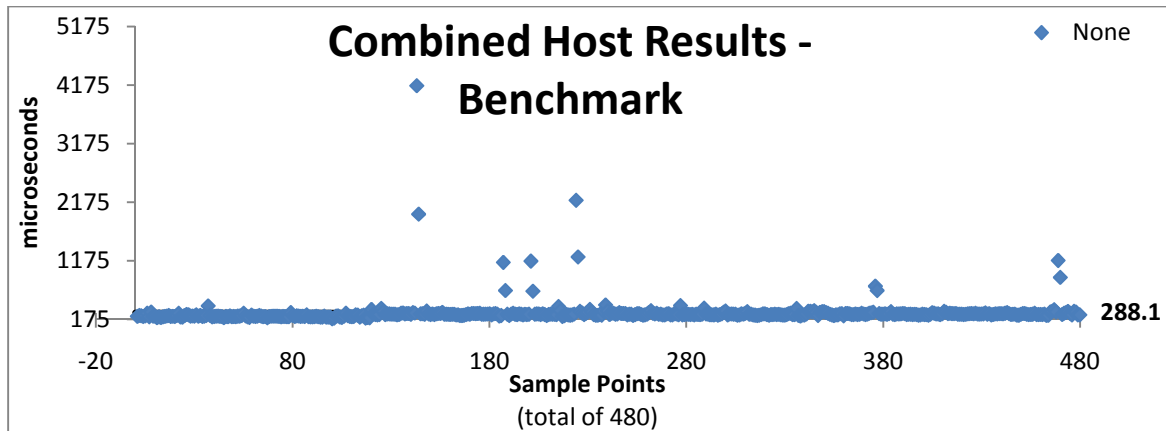
Figure 85: Combined Host Results - Benchmark
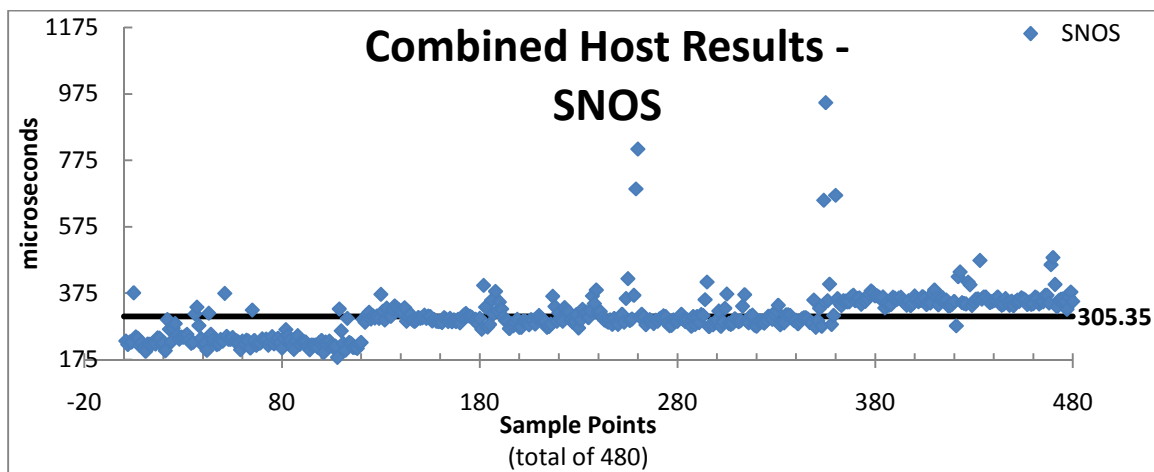
Figure 86: Combined Host Results – SNOS

The spread and variation within each trial for all the hosts combined is visually evident in Figure 87 because the SNOS trial's box plot is fatter meaning that the SNOS trial produces greater variance between results than the benchmark trial. Since the network load differed between each host, the results do not correlate to any usable statistical information. An ANOVA table is not provided to distract from the statistical relevance already determined by comparing the results individually for each host in the previous sections.
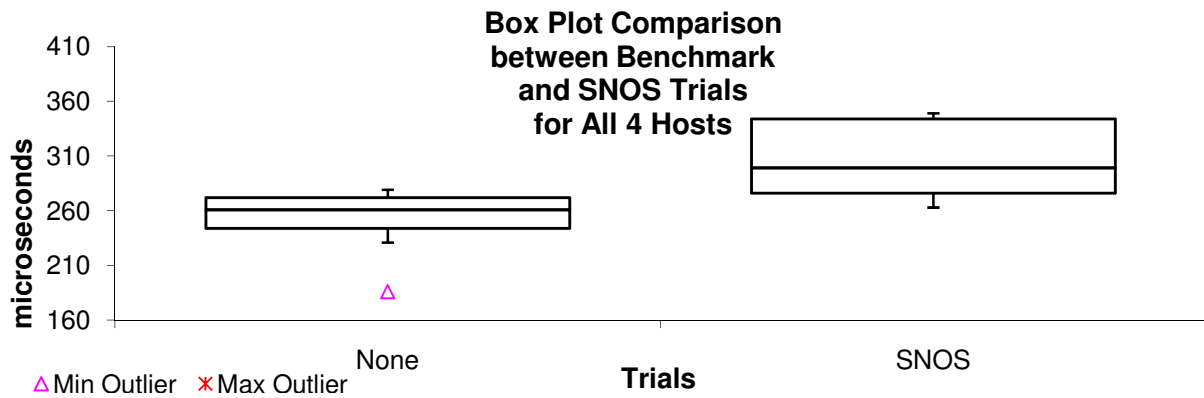
Figure 87: Box Plot – Combined Host Results

5.6   Additional Benefits of SNOS

To defeat the fingerprinting techniques employed by Nmap and Nessus, SNOS
blocks incoming ICMP timestamp requests and SMB Null Session requests. A Windows
operating system uses little endian format and so the ICMP timestamp response generated
by Windows is in little endian format. Nessus sometimes accurately identifies the
Windows host by determining if the most significant bit is set in the timestamp reply –
indicating the little endian format. By blocking ICMP timestamp requests, SNOS
prevents a remote user from identifying the date and time of the host which are
sometimes used to help coordinate further attacks against the host.

By responding to an SMB Null Session request, the host inherently identifies
itself as a Windows operating system. The SMB Null Session can also be used to
enumerate usernames on the host. Usernames identify the host operating system because
Windows uses predefined usernames. By blocking all SMB Null Session requests, SNOS
does not allow a remote user to enumerate the usernames of the host.

Blocking SMB Null Sessions can help block worm propagation throughout the
network. Worms sometimes use SMB to attack file shares using Null Sessions [Geb04].

131

Blocking Null Sessions on each host, by running SNOS, can block the propagation of

worms that use SMB Null Sessions to infect new hosts.

VI. Conclusion and Recommendations

6.1   Conclusion

A potential attacker spends a significant amount of time researching a target. By identifying the operating system running on a host, the attacker can determine possible vulnerabilities and methods of exploitation against that host. Most existing obfuscation research and programs focus on the Linux operating system. One of the few Windows-based obfuscation programs, OSfuscate, only focuses on defeating TCP/IP fingerprinting. All the current obfuscation programs modify a very small, finite, set of protocols. By only obfuscating one or two protocols, these programs do not effectively obfuscate the operating system of the host. A complete obfuscation process must occur in order to effectively defeat the wide range of fingerprinting techniques currently deployed.

The Systemic Network Obfuscation System (SNOS) obfuscates the Windows operating system. SNOS does not focus solely on the TCP and IP protocols but obfuscates protocols ranging from the Data Link layer to the Application layer in the OSI model. SNOS adheres to the principle that if the program does not obfuscate consistently and from all directions (in this case all layers of the OSI model) then the attacker would not be fooled. Obfuscation must be a process potentially involving all protocols within a network packet from a host.

The evaluation of the System Network Obfuscation System uses two metrics – the first to determine the obfuscation effectiveness of SNOS compared to OSfuscate and the benchmark, and the second to determine the network latency created by running SNOS on the host. Four host configurations are used to test the variance of each

protocol/service running based on the most common protocols found on a network and protocols identified as being used to fingerprint the underlying operating system of the host. The application serving up a particular protocol on a host is selected based on the most common Windows-based applications for each protocol. In the case of the HTTP web server, the top two most used applications are tested against – Apache and Microsoft IIS.

The network latency experiment uses the roundtrip time to determine the latency differences between the benchmark and SNOS trials. The Tcpreplay program injects identical network traffic to a host during both trials to eliminate the network traffic load as a possible cause of variance between the results.

The obfuscation effectiveness experiment produces 1,920 results and network latency experiment yields 960 results. The results of the obfuscation experiment are grouped according to the specific test and host. Each of the four hosts has a total of four fingerprinting tests run against it for each trial. The four tests result from two popular fingerprinting programs – Nmap and Nessus – and the results are separated according to the fingerprinting program and then subdivided according the test – OS Class or Service within each fingerprinting program.

The Systemic Network Obfuscation System obfuscates the underlying host's operating system 100% of the time against each test for each host. This success rate is achievable because of the finite amount of fingerprinting techniques which allowed the SNOS program to be created to defeat each identified technique. The benchmark, as expected, yields a 100% failure rate, meaning that each fingerprinting program is able to correctly identify the host's operating system as Windows. Fisher's Exact Test's

134

calculations and histogram figures show the obfuscation effectiveness of SNOS compared to the benchmark and OSfuscate. Out of the three trials, SNOS is the only trial, and therefore program, able defeat all the fingerprinting techniques used by Nmap and Nessus.

The results and analysis of the network latency experiment show that the SNOS trial produces greater variance, spread, within its results. With two of the hosts – web and workstation, the SNOS trial causes additional network latency as shown through the comparison of means, ANOVA and F-Test results. The remaining two hosts – Exchange and SharePoint produce mixed analytical results and do not provide a conclusive analysis regarding the cause of any additional network latency. So while SNOS accurately obfuscates the network packets for the host configurations used in these experiments, SNOS has mixed results whether or not the SNOS program causes additional statistically significant network latency for each host individually.

6.2   Host-Based Obfuscation Benefits

Obfuscation can provide another layer of defense for a host but should not be considered the primary security feature. Operating system service packs and application patches should be a top priority in securing a network along with an intrusion detection system and firewall that can block and detect anomalies. Host-based obfuscation should not be considered a replacement for network security devices but be considered a defense in depth mechanism.

By obfuscating host information, an attacker might spend additional time and resources trying to identify the host. As a result of this increased time, the attacker

135

might move on to another target or allow the fingerprinting attempts to be detected by increasing the amount of time and packets required to fingerprint a host [Rep08]. If host obfuscation is used throughout a LAN, either through a network-based obfuscator or having a similarly configured obfuscation program running on each host, each host will become anonymous among a group of anonymous other hosts [LiT08]. Revealing the host-level information makes an exploit easier and faster to use against a target host [Ber03]. Despite some claims, 'security-by-obscurity' methods can help thwart automated attacks [Sha04]. Most payloads delivered and executed through an exploit are operating system dependent, meaning a Windows exploit will not work for a Linux operating system. Once an operating system has been correctly identified default exploit attempts can be tried, including password guessing using default operating system specific administrative usernames.

Host obfuscation can limit the effectiveness of evasion attacks. An obfuscation technique that changes the TCP/IP stack implementation might affect the way the host re-orders fragmented packets. These changes to the TCP/IP stack can block evasion attacks aimed at traffic fragmentation [WSM04]. Obfuscation can be considered a security feature by providing additional defense in depth in an attempt to make host-level fingerprinting harder and less accurate.

6.3   Future Research

Systemic Network Obfuscation System obfuscatse IPv4 network packets. IPv6 is continuing to replace IPv4 as the need for additional Internet Protocol addresses increases throughout the world. Additional work improving SNOS should focus on implementing

136

IPv6 obfuscation. Implementing IPv6 should be fairly simple because only the IP header section of the SNOS program needs to be updated to detect and obfuscate IPv6 packets.

Along with IPv6, additional protocols, specifically at the application layer, could be included with SNOS for obfuscation. Although SNOS currently obfuscates the most common protocols found on a network, Windows operating systems enable a wide range of protocols by default including Remote Procedure Call (RPC), port map listening (port 135), and Active Directory related protocols and services. Recommended future work would be to increase the number of protocols SNOS is able to obfuscate.

SNOS runs as a user-level program instead of a kernel-level program. User-level programs are slower and more resource intensive than kernel-level programs. Additional research could be used to reprogram key components of how SNOS intercepts the traffic using the Winpkfilter driver in order to allow SNOS to run as a kernel-level program. By creating SNOS to run as a kernel-level program, the network latency caused by SNOS can be shortened because SNOS will not have to jump between user-level and kernel-level space to obfuscate the network traffic.

The Systemic Network Obfuscation System results in network packets that have identical protocol header field values between different hosts for the same protocols. In a sense, SNOS does traffic normalization indirectly because of the modifications SNOS makes to the various protocols. The custom, partially normalized network packets resulting from SNOS could be used to more accurately tune an intrusion detection system (IDS) or an intrusion prevention system (IPS). Additional research should focus on integrating SNOS-modified packets with an IDS and IPS devices.

An IPS, IDS, or even a firewall can be configured to flag or block packets that vary from the SNOS modified packets for each protocol. This process can be used to help detect and remove covert channels because the convert channel program alters a small portion of a packet. Rogue hosts, devices that are not supposed to be allowed on a particular network, can also be detected by comparing packets from a host to what the packet should look like after the SNOS modifications. Any difference between a SNOS modified packet and a captured packet from a host means that the host is not running SNOS and ideally all Windows devices on a particular network will utilize SNOS.

SNOS can also be extended to integrate a virtual TCP/IP stack implementation. Virtual TCP/IP stacks allow each service on a host to be running on a different host. SNOS can implement this feature by extending SNOS to assign specific MAC address and IP address combinations to a specific application running on the host, creating additional MAC and IP addresses as needed. SNOS would need to keep track of the MAC and IP address combinations and the application each combination matches within a database or lookup table. IP/Port hopping methods could also be employed as a part of this additional virtual TCP/IP functionality to create a dynamically changing view of the entire network infrastructure.

SNOS can be extended to work as a network-based obfuscation tool by setting up the Windows host to act as the default gateway for a network. If the Windows computer that SNOS is installed on is used as the network router then SNOS can be used to obfuscate all the packets entering and leaving a network as well.

SNOS can be combined and tested with other current obfuscation programs and techniques. IPSec modifies the packet information, typically all packet protocols above

138

the Transport layer in the TCP/IP model. IPSec encrypts the upper layer header fields and payload so that the packet data appears as random indistinguishable data. IPSec can allow only specific computers to communicate between each other on specific protocols by encrypting the traffic between each so that only the allowed computer can decrypt the packet. SNOS integration with IPSec would limit the protocols detected running on a host by making open ports that use IPSec to appear closed.

Additional work can combine IPsec and SNOS. SNOS can be modified to work in conjunction with the IPSec policies on each device, allowing SNOS to use the IPSec policies to determine which devices are allowed access to the host. SNOS can block each packet by simply ignoring/dropping requests sent to a host or by crafting custom response packets back to the original non-authorized host. Implementing IPSec within SNOS would allow for quick obfuscation of upper layer protocols and would be particularly beneficial for the communication between Active Directory domain controllers and/or Exchange servers. This obfuscation would limit the viewable ports and services available for an IPSec-SNOS host by all hosts except ones explicitly allowed through IPSec.

SNOS is the first obfuscation program to attempt to obfuscate all of the OSI layer protocols that make up a network packet. The results of the obfuscation effectiveness experiment provided analytical proof that an obfuscation process must obfuscate all layers of a packet in order to be effective as SNOS did.

Appendix A: SNOS Protocol Obfuscation List

Protocol	Obfuscation	Additional Information
HTTP Request	UserAgent field	
HTTP Response	Server field	
HTTP Response	X-Powered-By field	
HTTP Response	MicrosoftSharePointTeamServices field	
HTTP Response	MS-Authored-By field	
HTTP Response	DAV field	
HTTP Response	DASL field	
HTTP Response	Public field	
HTTP Response	Modify HTML content (Response codes: 400 & 401)	
IP	TTL field	
IP	IPID field	
IP	DF bit changed to 0	
IP	EnablePMTUDiscovery	Registry modification
IP	Recalculate IP checksum	
ICMP Request	Block Incoming ICMP Timestamp requests	
ICMP Reply	Increase payload size	
ICMP Reply	Modify payload data	
ICMP Reply	Recalculate ICMP checksum	
SMB Response	NativeOS field	
SMB Response	Natvie Lan Manager field	
SMB Response	Native Primary Domain field	
SMB Response	Flags	
SMB Response	Flags2 for AndX response	
SMB Request	Obfuscate Dialect Index Field	
SMB Request	Block incoming Null session requests	
SMB Request	Block data available to Null sessions	Registry modification
TCP	Acknowledgement # field (w/out a corresponding SYN packet)	
TCP	Window Size	Registry modification
TCP	Options	Registry modification
TCP	TcpUseRFC1122UrgentPointer	Registry modification
TCP	SackOpts	Registry modification
TCP	Recalculate TCP checksum	
Ethernet	MTU	Registry modification
TCP	Removed Option NOP, sack permitted	

SMTP Response	Email server banner (status 200)	
Browser Protocol	OS Major	
Browser Protocol	OS Minor	
UDP	Recalculate UDP checksum	
DHCP	Modified Option 60 value (MSFT 5.0)	
DHCP	Modified Option 55 values	
DHCP	Modified Bootp_flags	
SSH Request	Drop incoming packets from Protocol: SSHx-x-Nmap	
SSH Response	Modifiy SSH protocol to SSH-2.0-OpenSSH_3.8.1p1	
DNS Response	Obfuscate flag options	

Appendix B: Obfuscation Effectiveness Results

Run #: 1

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)91%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)91%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)91%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)91%	N	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 2

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)91%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)91%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)91%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)91%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 3

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)91%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)91%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)91%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)91%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N	N	N	N(445)

Run #: 4

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)91%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)92%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)91%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)91%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)91%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)92%	N	N	N(445)

Run #: 5

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)92%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)92%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)92%	N	N	N(445)

Run #: 6

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)95%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)95%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)95%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)95%	N	N	N(445)

Run #: 7

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 8

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)92%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)92%	N	N	N(445)

Run #: 9

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)94%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)93%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)93%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)93%	N	N	N(445)

Run #: 10

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)92%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	N	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 11

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)92%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)92%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)92%	N	N	N(445)

Run #: 12

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	N	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 13

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)94%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)93%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)94%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	N	Y(TCP/IP, SMB)	Y(445)	N(22)94%	N	N	N(445)

Run #: 14

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)89%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	N	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 15

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)92%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)92%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)92%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 16

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)92%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)92%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)94%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)94%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(139)	N(22)92%	N	N	N(445)

Run #: 17

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)92%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)92%	N	N	N(445)

Run #: 18

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,139)	N(25)92%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)92%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)92%	N	N	N(445)

Run #: 19

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)87%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)87%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)87%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)87%	N	N	N(445)

Run #: 20

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)93%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)89%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)91%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 21

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)92%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)87%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)87%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)93%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)87%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)92%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)87%	N	N	N(445)

Run #: 22

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	N	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 23

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)89%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	N	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 24

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)85%	N	N	N(445)

Run #: 25

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	N	Y(TCP/IP, SMB)	Y(445)	N(22)85%	N	N	N(445)

Run #: 26

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)85%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	N	Y(TCP/IP, SMB)	Y(445)	N	N	N	N(445)

Run #: 27

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)95%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)87%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)87%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	N	Y(TCP/IP, SMB)	Y(445)	N(22)94%	N	N	N(445)

Run #: 28

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)95%	N	N	N(445)
Sharepoint	Y(80)99%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)87%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)87%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)94%	N	N	N(445)

Run #: 29

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)85%	N	N	N(445)

Run #: 30

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)89%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)85%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	N	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 31

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	N	Y(TCP/IP, SMB)	Y(445)	N(22)85%	N	N	N(445)

Run #: 32

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)85%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)95%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)85%	N	N	N(445)

Run #: 33

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N	N	N	N(445)

Run #: 34

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)89%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)85%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)89%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)85%	N	N	N(445)

Run #: 35

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)89%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)85%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)88%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)85%	N	N	N(445)

Run #: 36

| | No Obfuscation | | | | OSfuscate | | | | SNOS | | | |
| | Nmap | | Nessus | | Nmap | | Nessus | | Nmap | | Nessus | |
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)89%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)85%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)98%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	N	Y(TCP/IP, SMB)	Y(445)	N(22)85%	N	N	N(445)

Run #: 37

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)91%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 38

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)85%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)89%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Workstation	Y(22)100%	N	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Run #: 39

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)85%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	N	Y(TCP/IP, SMB)	Y(445)	N(22)85%	N	N	N(445)

Run #: 40

	No Obfuscation				OSfuscate				SNOS			
	Nmap		Nessus		Nmap		Nessus		Nmap		Nessus	
	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services	OS Class	Services	OS Ident	Services
Exchange	Y(25)100%	Y(25,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	Y(25)90%	Y(25,53,80,445)	Y(TCP/IP, SMB, HTTP)	Y(25,80,445)	N(25)85%	N	N	N(445)
Sharepoint	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N(80)85%	N	N	N(445)
Web	Y(80)100%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	Y(80)90%	Y(80,445)	Y(TCP/IP, SMB, HTTP)	Y(80,445)	N	N	N	N(445)
Workstation	Y(22)100%	Y(445)	Y(TCP/IP, SMB)	Y(445)	Y(22)89%	Y(445)	Y(TCP/IP, SMB)	Y(445)	N(22)91%	N	N	N(445)

Appendix C: Network Latency Results

Run #: 1

	No Obfuscation					SNOS				
	Min	Max	Mean	SD	CI	Min	Max	Mean	SD	CI
Exchange	205.69	301.87	237.35	27.30	20.30	200.83	376.29	237.10	45.10	33.76
Sharepoint	252.30	364.02	274.14	31.36	23.32	288.46	372.03	309.43	21.93	16.30
Web	250.87	289.19	2269.79	10.90	8.11	281.67	319.79	296.42	11.79	8.77
Workstation	255.35	278.37	265.10	7.33	5.45	337.80	368.70	354.28	8.72	6.48

Run #: 2

	No Obfuscation					SNOS				
	Min	Max	Mean	SD	CI	Min	Max	Mean	SD	CI
Exchange	211.00	278.00	229.67	17.67	13.15	202.69	295.59	234.50	24.68	18.35
Sharepoint	252.00	4164.58	738.76	1184.19	880.54	290.63	336.79	318.85	14.26	10.60
Web	242.20	321.71	269.13	21.55	16.03	280.05	809.33	393.87	172.77	128.47
Workstation	263.29	741.81	3448.84	167.25	124.36	341.85	382.56	363.10	11.33	8.43

Run #: 3

	No Obfuscation					SNOS				
	Min	Max	Mean	SD	CI	Min	Max	Mean	SD	CI
Exchange	223.58	258.93	238.97	11.48	8.54	224.83	313.18	251.89	25.52	18.97
Sharepoint	257.00	318.00	274.17	17.00	12.64	289.70	312.16	300.58	6.94	5.16
Web	249.52	277.73	261.38	9.51	7.07	277.00	307.00	293.67	8.90	6.62
Workstation	258.00	277.00	265.08	6.16	4.58	331.00	363.00	349.75	10.86	8.08

Run #: 4

	No Obfuscation					SNOS				
	Min	Max	Mean	SD	CI	Min	Max	Mean	SD	CI
Exchange	204.00	407.00	241.00	53.02	39.42	203.00	333.00	247.25	40.64	30.22
Sharepoint	239.97	263.77	254.82	8.95	6.65	287.00	302.00	293.08	4.78	3.55
Web	254.00	410.00	271.83	43.66	32.47	276.00	312.00	292.92	10.51	7.81
Workstation	246.00	291.00	262.92	12.75	9.48	338.00	367.00	352.25	9.23	6.86

Run #: 5

	No Obfuscation					SNOS				
	Min	Max	Mean	SD	CI	Min	Max	Mean	SD	CI
Exchange	209.00	282.00	232.42	18.72	13.92	205.00	375.00	244.17	42.39	31.52
Sharepoint	261.00	278.00	269.42	6.05	4.50	283.00	314.00	300.08	9.04	6.72
Web	249.00	365.00	274.92	34.57	25.70	275.00	409.00	306.25	39.14	29.11
Workstation	257.00	312.00	274.92	15.58	11.59	337.00	385.00	354.75	14.67	10.91

Run #: 6

	No Obfuscation					SNOS				
	Min	Max	Mean	SD	CI	Min	Max	Mean	SD	CI
Exchange	200.89	243.76	226.62	9.81	7.30	210.00	325.00	236.00	29.22	21.73
Sharepoint	233.00	1150.00	367.08	273.43	203.32	267.00	399.00	325.00	41.12	30.57
Web	246.00	305.00	265.58	15.24	11.33	276.00	373.00	301.25	27.70	20.60
Workstation	259.00	286.00	270.83	7.31	5.43	277.00	439.00	365.67	45.08	33.52

Run #: 7

	No Obfuscation					SNOS				
	Min	Max	Mean	SD	CI	Min	Max	Mean	SD	CI
Exchange	219.00	290.00	231.75	19.50	14.50	210.00	266.00	234.00	14.98	11.14
Sharepoint	253.00	1170.00	371.17	275.97	205.21	269.00	297.00	284.25	9.19	6.83
Web	242.00	268.00	255.17	8.86	6.59	276.00	371.00	300.92	27.37	20.35
Workstation	253.00	289.00	269.50	9.79	7.28	339.00	474.00	363.83	35.58	26.45

Run #: 8

	No Obfuscation					SNOS				
	Min	Max	Mean	SD	CI	Min	Max	Mean	SD	CI
Exchange	212.00	253.00	224.75	10.96	8.15	205.00	248.00	222.58	11.90	8.85
Sharepoint	239.00	395.00	275.92	42.40	31.53	273.00	308.00	289.75	11.22	8.34
Web	248.00	361.00	268.42	30.68	22.81	81.00	339.00	304.00	16.56	12.31
Workstation	247.00	274.00	261.75	8.35	6.21	337.00	361.00	350.67	7.81	5.81

Run #: 9

	No Obfuscation					SNOS				
	Min	*Max*	*Mean*	*SD*	*CI*	*Min*	*Max*	*Mean*	*SD*	*CI*
Exchange	186.00	278.00	221.33	22.94	17.05	182.00	233.00	216.50	14.64	10.89
Sharepoint	224.00	2210.00	503.50	607.28	451.56	288.00	366.00	312.00	24.39	18.14
Web	232.75	324.89	277.83	33.11	24.62	283.00	308.00	295.17	7.49	5.57
Workstation	241.00	338.00	276.33	30.25	22.49	341.00	369.00	352.08	10.52	7.82

Run #: 10

	No Obfuscation					SNOS				
	Min	*Max*	*Mean*	*SD*	*CI*	*Min*	*Max*	*Mean*	*SD*	*CI*
Exchange	195.00	347.00	235.42	39.78	29.58	203.00	328.00	237.50	39.11	29.08
Sharepoint	244.00	422.00	294.25	52.40	38.96	271.00	385.00	322.92	30.82	22.92
Web	236.00	305.00	267.58	23.39	17.39	273.00	949.00	428.83	214.07	159.18
Workstation	250.00	1180.00	404.58	301.89	224.48	327.00	483.00	376.58	48.77	36.26

Appendix D: Nmap Operating System Identification

	Run	No Obfuscation			OSfuscate			SNOS		
		OS	Services	Combined	OS	Services	Combined	OS	Services	Combined
Exchange	1	Y	Y	Y	Y	Y	Y	N	N	N
	2	Y	Y	Y	Y	Y	Y	N	N	N
	3	Y	Y	Y	Y	Y	Y	N	N	N
	4	Y	Y	Y	Y	Y	Y	N	N	N
	5	Y	Y	Y	Y	Y	Y	N	N	N
	6	Y	Y	Y	Y	Y	Y	N	N	N
	7	Y	Y	Y	Y	Y	Y	N	N	N
	8	Y	Y	Y	Y	Y	Y	N	N	N
	9	Y	Y	Y	Y	Y	Y	N	N	N
	10	Y	Y	Y	Y	Y	Y	N	N	N
	11	Y	Y	Y	Y	Y	Y	N	N	N
	12	Y	Y	Y	Y	Y	Y	N	N	N
	13	Y	Y	Y	Y	Y	Y	N	N	N
	14	Y	Y	Y	Y	Y	Y	N	N	N
	15	Y	Y	Y	Y	Y	Y	N	N	N
	16	Y	Y	Y	Y	Y	Y	N	N	N
	17	Y	Y	Y	Y	Y	Y	N	N	N
	18	Y	Y	Y	Y	Y	Y	N	N	N
	19	Y	Y	Y	Y	Y	Y	N	N	N
	20	Y	Y	Y	Y	Y	Y	N	N	N
	21	Y	Y	Y	Y	Y	Y	N	N	N
	22	Y	Y	Y	Y	Y	Y	N	N	N
	23	Y	Y	Y	Y	Y	Y	N	N	N
	24	Y	Y	Y	Y	Y	Y	N	N	N
	25	Y	Y	Y	Y	Y	Y	N	N	N
	26	Y	Y	Y	Y	Y	Y	N	N	N
	27	Y	Y	Y	Y	Y	Y	N	N	N
	28	Y	Y	Y	Y	Y	Y	N	N	N
	29	Y	Y	Y	Y	Y	Y	N	N	N
	30	Y	Y	Y	Y	Y	Y	N	N	N
	31	Y	Y	Y	Y	Y	Y	N	N	N
	32	Y	Y	Y	Y	Y	Y	N	N	N
	33	Y	Y	Y	Y	Y	Y	N	N	N
	34	Y	Y	Y	Y	Y	Y	N	N	N
	35	Y	Y	Y	Y	Y	Y	N	N	N
	36	Y	Y	Y	Y	Y	Y	N	N	N
	37	Y	Y	Y	Y	Y	Y	N	N	N
	38	Y	Y	Y	Y	Y	Y	N	N	N
	39	Y	Y	Y	Y	Y	Y	N	N	N

	40	Y	Y	Y	Y	Y	Y	N	N	N
SharePoint	1	Y	Y	Y	Y	Y	Y	N	N	N
	2	Y	Y	Y	Y	Y	Y	N	N	N
	3	Y	Y	Y	Y	Y	Y	N	N	N
	4	Y	Y	Y	Y	Y	Y	N	N	N
	5	Y	Y	Y	Y	Y	Y	N	N	N
	6	Y	Y	Y	Y	Y	Y	N	N	N
	7	Y	Y	Y	Y	Y	Y	N	N	N
	8	Y	Y	Y	Y	Y	Y	N	N	N
	9	Y	Y	Y	Y	Y	Y	N	N	N
	10	Y	Y	Y	Y	Y	Y	N	N	N
	11	Y	Y	Y	Y	Y	Y	N	N	N
	12	Y	Y	Y	Y	Y	Y	N	N	N
	13	Y	Y	Y	Y	Y	Y	N	N	N
	14	Y	Y	Y	Y	Y	Y	N	N	N
	15	Y	Y	Y	Y	Y	Y	N	N	N
	16	Y	Y	Y	Y	Y	Y	N	N	N
	17	Y	Y	Y	Y	Y	Y	N	N	N
	18	Y	Y	Y	Y	Y	Y	N	N	N
	19	Y	Y	Y	Y	Y	Y	N	N	N
	20	Y	Y	Y	Y	Y	Y	N	N	N
	21	Y	Y	Y	Y	Y	Y	N	N	N
	22	Y	Y	Y	Y	Y	Y	N	N	N
	23	Y	Y	Y	Y	Y	Y	N	N	N
	24	Y	Y	Y	Y	Y	Y	N	N	N
	25	Y	Y	Y	Y	Y	Y	N	N	N
	26	Y	Y	Y	Y	Y	Y	N	N	N
	27	Y	Y	Y	Y	Y	Y	N	N	N
	28	Y	Y	Y	Y	Y	Y	N	N	N
	29	Y	Y	Y	Y	Y	Y	N	N	N
	30	Y	Y	Y	Y	Y	Y	N	N	N
	31	Y	Y	Y	Y	Y	Y	N	N	N
	32	Y	Y	Y	Y	Y	Y	N	N	N
	33	Y	Y	Y	Y	Y	Y	N	N	N
	34	Y	Y	Y	Y	Y	Y	N	N	N
	35	Y	Y	Y	Y	Y	Y	N	N	N
	36	Y	Y	Y	Y	Y	Y	N	N	N
	37	Y	Y	Y	Y	Y	Y	N	N	N
	38	Y	Y	Y	Y	Y	Y	N	N	N
	39	Y	Y	Y	Y	Y	Y	N	N	N
	40	Y	Y	Y	Y	Y	Y	N	N	N
Web	1	Y	Y	Y	Y	Y	Y	N	N	N
	2	Y	Y	Y	Y	Y	Y	N	N	N
	3	Y	Y	Y	Y	Y	Y	N	N	N

4	Y	Y	Y	Y	Y	Y	N	N	N
5	Y	Y	Y	Y	Y	Y	N	N	N
6	Y	Y	Y	Y	Y	Y	N	N	N
7	Y	Y	Y	Y	Y	Y	N	N	N
8	Y	Y	Y	Y	Y	Y	N	N	N
9	Y	Y	Y	Y	Y	Y	N	N	N
10	Y	Y	Y	Y	Y	Y	N	N	N
11	Y	Y	Y	Y	Y	Y	N	N	N
12	Y	Y	Y	Y	Y	Y	N	N	N
13	Y	Y	Y	Y	Y	Y	N	N	N
14	Y	Y	Y	Y	Y	Y	N	N	N
15	Y	Y	Y	Y	Y	Y	N	N	N
16	Y	Y	Y	Y	Y	Y	N	N	N
17	Y	Y	Y	Y	Y	Y	N	N	N
18	Y	Y	Y	Y	Y	Y	N	N	N
19	Y	Y	Y	Y	Y	Y	N	N	N
20	Y	Y	Y	Y	Y	Y	N	N	N
21	Y	Y	Y	Y	Y	Y	N	N	N
22	Y	Y	Y	Y	Y	Y	N	N	N
23	Y	Y	Y	Y	Y	Y	N	N	N
24	Y	Y	Y	Y	Y	Y	N	N	N
25	Y	Y	Y	Y	Y	Y	N	N	N
26	Y	Y	Y	Y	Y	Y	N	N	N
27	Y	Y	Y	Y	Y	Y	N	N	N
28	Y	Y	Y	Y	Y	Y	N	N	N
29	Y	Y	Y	Y	Y	Y	N	N	N
30	Y	Y	Y	Y	Y	Y	N	N	N
31	Y	Y	Y	Y	Y	Y	N	N	N
32	Y	Y	Y	Y	Y	Y	N	N	N
33	Y	Y	Y	Y	Y	Y	N	N	N
34	Y	Y	Y	Y	Y	Y	N	N	N
35	Y	Y	Y	Y	Y	Y	N	N	N
36	Y	Y	Y	Y	Y	Y	N	N	N
37	Y	Y	Y	Y	Y	Y	N	N	N
38	Y	Y	Y	Y	Y	Y	N	N	N
39	Y	Y	Y	Y	Y	Y	N	N	N
40	Y	Y	Y	Y	Y	Y	N	N	N
Workstation 1	Y	N	Y	Y	N	Y	N	N	N
2	Y	Y	Y	Y	Y	Y	N	N	N
3	Y	Y	Y	Y	Y	Y	N	N	N
4	Y	Y	Y	Y	Y	Y	N	N	N
5	Y	Y	Y	Y	Y	Y	N	N	N
6	Y	Y	Y	Y	Y	Y	N	N	N
7	Y	N	Y	Y	Y	Y	N	N	N

8	Y	Y	Y	Y	Y	Y	N	N	N
9	Y	Y	Y	Y	Y	Y	N	N	N
10	Y	N	Y	Y	N	Y	N	N	N
11	Y	Y	Y	Y	Y	Y	N	N	N
12	Y	N	Y	Y	N	Y	N	N	N
13	Y	Y	Y	Y	N	Y	N	N	N
14	Y	N	Y	Y	N	Y	N	N	N
15	Y	Y	Y	Y	Y	Y	N	N	N
16	Y	Y	Y	Y	Y	Y	N	N	N
17	Y	N	Y	Y	Y	Y	N	N	N
18	Y	Y	Y	Y	Y	Y	N	N	N
19	Y	Y	Y	Y	Y	Y	N	N	N
20	Y	Y	Y	Y	Y	Y	N	N	N
21	Y	Y	Y	Y	Y	Y	N	N	N
22	Y	N	Y	Y	N	Y	N	N	N
23	Y	Y	Y	Y	N	Y	N	N	N
24	Y	Y	Y	Y	Y	Y	N	N	N
25	Y	N	Y	Y	N	Y	N	N	N
26	Y	Y	Y	Y	N	Y	N	N	N
27	Y	Y	Y	Y	N	Y	N	N	N
28	Y	N	Y	Y	Y	Y	N	N	N
29	Y	Y	Y	Y	Y	Y	N	N	N
30	Y	Y	Y	Y	N	Y	N	N	N
31	Y	N	Y	Y	N	Y	N	N	N
32	Y	N	Y	Y	Y	Y	N	N	N
33	Y	Y	Y	Y	Y	Y	N	N	N
34	Y	N	Y	Y	Y	Y	N	N	N
35	Y	N	Y	Y	Y	Y	N	N	N
36	Y	Y	Y	Y	N	Y	N	N	N
37	Y	N	Y	Y	Y	Y	N	N	N
38	Y	N	Y	Y	Y	Y	N	N	N
39	Y	Y	Y	Y	N	Y	N	N	N
40	Y	Y	Y	Y	Y	Y	N	N	N

Total   **160**

Appendix E: Nmap Operating System Identification Histograms

Nmap Operating System Identification - Exchange
(Combined Test Results)

Legend: □ None ■ OSfuscate (yellow) ■ Snos

Number of Times Nmap believed the Host was running Windows Operating System

Nmap Operating System Identification - SharePoint
(Combined Test Results)

Legend: □ None ■ OSfuscate (yellow) ■ Snos

Number of Times Nmap believed the Host was running Windows Operating System

Nmap Operating System Identification - Web

(Combined Test Results)

Frequency Count (total of 40 per trial)

Number of Times Nmap believed the Host was running Windows Operating System

Legend: None, OSfuscate, Snos

Nmap Operating System Identification - Workstation

(Combined Test Results)

Frequency Count (total of 40 per trial)

Number of Times Nmap believed the Host was running Windows Operating System

Legend: None, OSfuscate, Snos

160

Appendix F: Nessus Operating System Identification

	Run	No Obfuscation			OSfuscate			SNOS		
		OS	Services	Combined	OS	Services	Combined	OS	Services	Combined
Exchange	1	Y	Y	Y	Y	Y	Y	N	N	N
	2	Y	Y	Y	Y	Y	Y	N	N	N
	3	Y	Y	Y	Y	Y	Y	N	N	N
	4	Y	Y	Y	Y	Y	Y	N	N	N
	5	Y	Y	Y	Y	Y	Y	N	N	N
	6	Y	Y	Y	Y	Y	Y	N	N	N
	7	Y	Y	Y	Y	Y	Y	N	N	N
	8	Y	Y	Y	Y	Y	Y	N	N	N
	9	Y	Y	Y	Y	Y	Y	N	N	N
	10	Y	Y	Y	Y	Y	Y	N	N	N
	11	Y	Y	Y	Y	Y	Y	N	N	N
	12	Y	Y	Y	Y	Y	Y	N	N	N
	13	Y	Y	Y	Y	Y	Y	N	N	N
	14	Y	Y	Y	Y	Y	Y	N	N	N
	15	Y	Y	Y	Y	Y	Y	N	N	N
	16	Y	Y	Y	Y	Y	Y	N	N	N
	17	Y	Y	Y	Y	Y	Y	N	N	N
	18	Y	Y	Y	Y	Y	Y	N	N	N
	19	Y	Y	Y	Y	Y	Y	N	N	N
	20	Y	Y	Y	Y	Y	Y	N	N	N
	21	Y	Y	Y	Y	Y	Y	N	N	N
	22	Y	Y	Y	Y	Y	Y	N	N	N
	23	Y	Y	Y	Y	Y	Y	N	N	N
	24	Y	Y	Y	Y	Y	Y	N	N	N
	25	Y	Y	Y	Y	Y	Y	N	N	N
	26	Y	Y	Y	Y	Y	Y	N	N	N
	27	Y	Y	Y	Y	Y	Y	N	N	N
	28	Y	Y	Y	Y	Y	Y	N	N	N
	29	Y	Y	Y	Y	Y	Y	N	N	N
	30	Y	Y	Y	Y	Y	Y	N	N	N
	31	Y	Y	Y	Y	Y	Y	N	N	N
	32	Y	Y	Y	Y	Y	Y	N	N	N
	33	Y	Y	Y	Y	Y	Y	N	N	N
	34	Y	Y	Y	Y	Y	Y	N	N	N
	35	Y	Y	Y	Y	Y	Y	N	N	N
	36	Y	Y	Y	Y	Y	Y	N	N	N
	37	Y	Y	Y	Y	Y	Y	N	N	N
	38	Y	Y	Y	Y	Y	Y	N	N	N
	39	Y	Y	Y	Y	Y	Y	N	N	N

	40	Y	Y	Y	Y	Y	Y	N	N	N
SharePoint	**1**	Y	Y	Y	Y	Y	Y	N	N	N
	2	Y	Y	Y	Y	Y	Y	N	N	N
	3	Y	Y	Y	Y	Y	Y	N	N	N
	4	Y	Y	Y	Y	Y	Y	N	N	N
	5	Y	Y	Y	Y	Y	Y	N	N	N
	6	Y	Y	Y	Y	Y	Y	N	N	N
	7	Y	Y	Y	Y	Y	Y	N	N	N
	8	Y	Y	Y	Y	Y	Y	N	N	N
	9	Y	Y	Y	Y	Y	Y	N	N	N
	10	Y	Y	Y	Y	Y	Y	N	N	N
	11	Y	Y	Y	Y	Y	Y	N	N	N
	12	Y	Y	Y	Y	Y	Y	N	N	N
	13	Y	Y	Y	Y	Y	Y	N	N	N
	14	Y	Y	Y	Y	Y	Y	N	N	N
	15	Y	Y	Y	Y	Y	Y	N	N	N
	16	Y	Y	Y	Y	Y	Y	N	N	N
	17	Y	Y	Y	Y	Y	Y	N	N	N
	18	Y	Y	Y	Y	Y	Y	N	N	N
	19	Y	Y	Y	Y	Y	Y	N	N	N
	20	Y	Y	Y	Y	Y	Y	N	N	N
	21	Y	Y	Y	Y	Y	Y	N	N	N
	22	Y	Y	Y	Y	Y	Y	N	N	N
	23	Y	Y	Y	Y	Y	Y	N	N	N
	24	Y	Y	Y	Y	Y	Y	N	N	N
	25	Y	Y	Y	Y	Y	Y	N	N	N
	26	Y	Y	Y	Y	Y	Y	N	N	N
	27	Y	Y	Y	Y	Y	Y	N	N	N
	28	Y	Y	Y	Y	Y	Y	N	N	N
	29	Y	Y	Y	Y	Y	Y	N	N	N
	30	Y	Y	Y	Y	Y	Y	N	N	N
	31	Y	Y	Y	Y	Y	Y	N	N	N
	32	Y	Y	Y	Y	Y	Y	N	N	N
	33	Y	Y	Y	Y	Y	Y	N	N	N
	34	Y	Y	Y	Y	Y	Y	N	N	N
	35	Y	Y	Y	Y	Y	Y	N	N	N
	36	Y	Y	Y	Y	Y	Y	N	N	N
	37	Y	Y	Y	Y	Y	Y	N	N	N
	38	Y	Y	Y	Y	Y	Y	N	N	N
	39	Y	Y	Y	Y	Y	Y	N	N	N
	40	Y	Y	Y	Y	Y	Y	N	N	N
Web	**1**	Y	Y	Y	Y	Y	Y	N	N	N
	2	Y	Y	Y	Y	Y	Y	N	N	N
	3	Y	Y	Y	Y	Y	Y	N	N	N

4	Y	Y	Y	Y	Y	Y	N	N	N
5	Y	Y	Y	Y	Y	Y	N	N	N
6	Y	Y	Y	Y	Y	Y	N	N	N
7	Y	Y	Y	Y	Y	Y	N	N	N
8	Y	Y	Y	Y	Y	Y	N	N	N
9	Y	Y	Y	Y	Y	Y	N	N	N
10	Y	Y	Y	Y	Y	Y	N	N	N
11	Y	Y	Y	Y	Y	Y	N	N	N
12	Y	Y	Y	Y	Y	Y	N	N	N
13	Y	Y	Y	Y	Y	Y	N	N	N
14	Y	Y	Y	Y	Y	Y	N	N	N
15	Y	Y	Y	Y	Y	Y	N	N	N
16	Y	Y	Y	Y	Y	Y	N	N	N
17	Y	Y	Y	Y	Y	Y	N	N	N
18	Y	Y	Y	Y	Y	Y	N	N	N
19	Y	Y	Y	Y	Y	Y	N	N	N
20	Y	Y	Y	Y	Y	Y	N	N	N
21	Y	Y	Y	Y	Y	Y	N	N	N
22	Y	Y	Y	Y	Y	Y	N	N	N
23	Y	Y	Y	Y	Y	Y	N	N	N
24	Y	Y	Y	Y	Y	Y	N	N	N
25	Y	Y	Y	Y	Y	Y	N	N	N
26	Y	Y	Y	Y	Y	Y	N	N	N
27	Y	Y	Y	Y	Y	Y	N	N	N
28	Y	Y	Y	Y	Y	Y	N	N	N
29	Y	Y	Y	Y	Y	Y	N	N	N
30	Y	Y	Y	Y	Y	Y	N	N	N
31	Y	Y	Y	Y	Y	Y	N	N	N
32	Y	Y	Y	Y	Y	Y	N	N	N
33	Y	Y	Y	Y	Y	Y	N	N	N
34	Y	Y	Y	Y	Y	Y	N	N	N
35	Y	Y	Y	Y	Y	Y	N	N	N
36	Y	Y	Y	Y	Y	Y	N	N	N
37	Y	Y	Y	Y	Y	Y	N	N	N
38	Y	Y	Y	Y	Y	Y	N	N	N
39	Y	Y	Y	Y	Y	Y	N	N	N
40	Y	Y	Y	Y	Y	Y	N	N	N
Workstation **1**	Y	Y	Y	Y	Y	Y	N	N	N
2	Y	Y	Y	Y	Y	Y	N	N	N
3	Y	Y	Y	Y	Y	Y	N	N	N
4	Y	Y	Y	Y	Y	Y	N	N	N
5	Y	Y	Y	Y	Y	Y	N	N	N
6	Y	Y	Y	Y	Y	Y	N	N	N
7	Y	Y	Y	Y	Y	Y	N	N	N

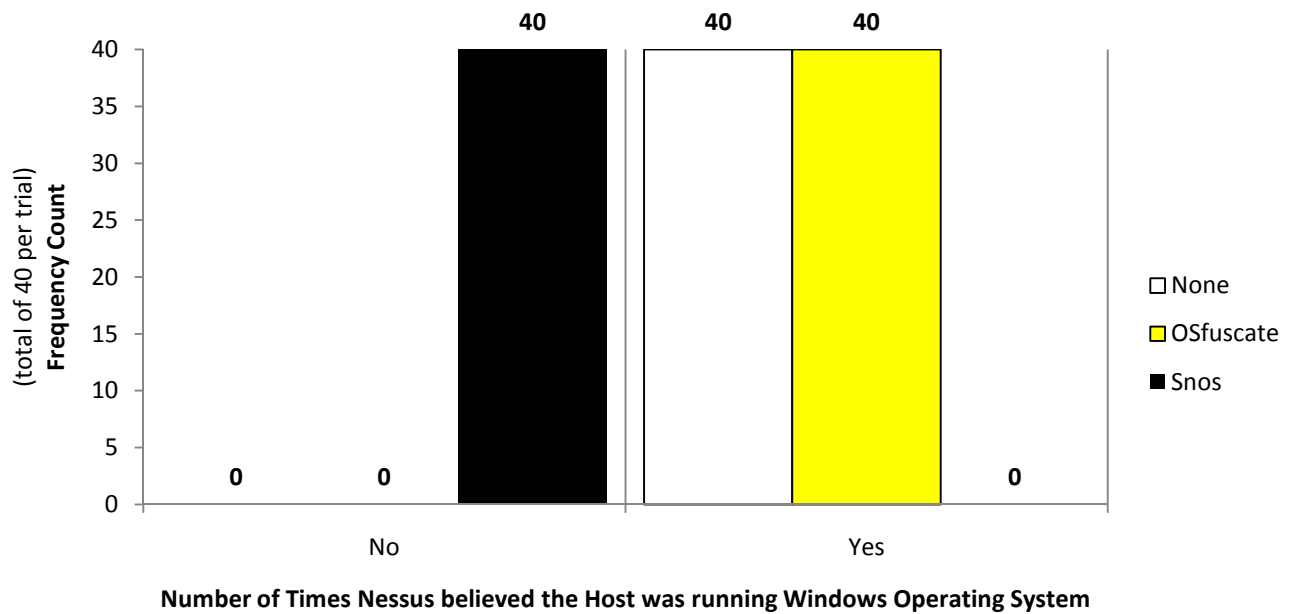|								
8	Y	Y	Y	Y	Y	Y	N	N	N
9	Y	Y	Y	Y	Y	Y	N	N	N
10	Y	Y	Y	Y	Y	Y	N	N	N
11	Y	Y	Y	Y	Y	Y	N	N	N
12	Y	Y	Y	Y	Y	Y	N	N	N
13	Y	Y	Y	Y	Y	Y	N	N	N
14	Y	Y	Y	Y	Y	Y	N	N	N
15	Y	Y	Y	Y	Y	Y	N	N	N
16	Y	Y	Y	Y	Y	Y	N	N	N
17	Y	Y	Y	Y	Y	Y	N	N	N
18	Y	Y	Y	Y	Y	Y	N	N	N
19	Y	Y	Y	Y	Y	Y	N	N	N
20	Y	Y	Y	Y	Y	Y	N	N	N
21	Y	Y	Y	Y	Y	Y	N	N	N
22	Y	Y	Y	Y	Y	Y	N	N	N
23	Y	Y	Y	Y	Y	Y	N	N	N
24	Y	Y	Y	Y	Y	Y	N	N	N
25	Y	Y	Y	Y	Y	Y	N	N	N
26	Y	Y	Y	Y	Y	Y	N	N	N
27	Y	Y	Y	Y	Y	Y	N	N	N
28	Y	Y	Y	Y	Y	Y	N	N	N
29	Y	Y	Y	Y	Y	Y	N	N	N
30	Y	Y	Y	Y	Y	Y	N	N	N
31	Y	Y	Y	Y	Y	Y	N	N	N
32	Y	Y	Y	Y	Y	Y	N	N	N
33	Y	Y	Y	Y	Y	Y	N	N	N
34	Y	Y	Y	Y	Y	Y	N	N	N
35	Y	Y	Y	Y	Y	Y	N	N	N
36	Y	Y	Y	Y	Y	Y	N	N	N
37	Y	Y	Y	Y	Y	Y	N	N	N
38	Y	Y	Y	Y	Y	Y	N	N	N
39	Y	Y	Y	Y	Y	Y	N	N	N
40	Y	Y	Y	Y	Y	Y	N	N	N

Total 160

Appendix G: Nessus Operating System Identification Histograms

Nessus Operating System Identification - Exchange
(Combined Test Results)

40   40   40

(total of 40 per trial)
Frequency Count

- □ None
- □ OSfuscate
- ■ Snos

0   0   0

No   Yes

Number of Times Nessus believed the Host was running Windows Operating System

Nessus Operating System Identification - SharePoint
(Combined Test Results)

40   40   40

(total of 40 per trial)
Frequency Count

- □ None
- □ OSfuscate
- ■ Snos

0   0   0

No   Yes

Number of Times Nessus believed the Host was running Windows Operating System

Nessus Operating System Identification - Web
(Combined Test Results)

Frequency Count (total of 40 per trial) vs Number of Times Nessus believed the Host was running Windows Operating System

Legend: None, OSfuscate, Snos

Nessus Operating System Identification - Workstation
(Combined Test Results)

Frequency Count (total of 40 per trial) vs Number of Times Nessus believed the Host was running Windows Operating System

Legend: None, OSfuscate, Snos

166

Bibliography

[All07]   Jon Mark Allen, "OS and Application Fingerprinting Techniques,"
            September 2007. [Online]. Available:
            http://www.sans.org/reading_room/whitepapers/protocols/os-application-
            fingerprinting-techniques_1891 [Accessed: March 4, 2011].

[Ark01]   Ofir Arkin, "ICMP Usage in Scanning," June 2001. [Online]. Available:
            http://www.net-security.org/article.php?id=54 [Accessed: May 22, 2010].

[Ark02]   Ofir Arkin, "A Remote Active OS Fingerprinting tool using ICMP."
            [Online]. Available: http://ofirarkin.files.wordpress.com/2008/11/login.pdf
            [Accessed: June 1, 2010].

[ASF11]   Apache Software Foundation, "Using Apache HTTP Server on Microsoft
            Windows." [Online]. Available:
            http://httpd.apache.org/docs/2.2/platform/windows.html [Accessed: January
            26, 2011].

[Bar52]   G. Barkas, *The Camouflage Story.* Cassell & Co. Ltd, 1952.

[Bec01]   Rob Beck, "Passive-aggressive resistance: OS Fingerprint evasion," in
            Linux Journal, vol. 2001, no. 89, pp. 1, 2001.

[Ber03]   David Barroso Berrueta, "A Practical approach for defeating Nmap OS-
            Fingerprinting." [Online]. Available:
            http://www.infosecwriters.com/text_resources/pdf/nmap.pdf [Accessed:
            March 21, 2010].

[BHP07]   Genevieve Bartlett, John Heidemann and Christos Papadopoulos,
            "Understanding passive and active service discovery," in Proceedings of the
            7th ACM SIGCOMM conference on Internet measurement, pp. 57-70, 2007.

[Cre08]   A. Crenshaw, "OSfuscate: Change your Windows OS TCP/IP fingerprint to
            confuse P0f, NetworkMiner, Ettercap, Nmap, and other OS detection tools."
            [Online]. Available:
            http://www.irongeek.com/i.php?page=security/osfuscate-change-your-
            windows-os-tcp-ip-fingerprint-to-confuse-p0f-networkminer-ettercap-nmap-
            and-other-os-detection-tools [Accessed: March 17, 2010].

[DFM06]   Holger Dreger, Anja Feldmann, Michael Mai, Vern Paxson and Robin
            Sommer, "Dynamic Application-Layer Protocol Analysis for Network
            Intrusion Detection." [Online]. Available:

http://www.icir.org/robin/papers/usenix06.pdf [Accessed: February 10, 2011].

[Fer08]   Ferris Research, Inc, "Email Products: Market Shares, Versions Deployed, Migrations, and Software Cost." [Online]. Available: http://www.ferris.com/?file_id=2008/02/319498_764-Email-Census Subset-04mc.pdf&ela=367317&stp=762212 [Accessed: July 23, 2010].

[Fra09]   Carl Frappaolo, "State of the Market: Microsoft Sharepoint." [Online]. Available: http://www.aiim.org/pdfdocuments/35927.pdf [Accessed: February 16, 2011].

[Fyo02]   Fyodor Yarochkin, "Remote OS Detection." [Online]. Availabe: http://nmap.org/book/osdetect-methods.html#osdetect-cd [Accessed: March 11, 2010].

[Geb04]   Glenn Gebhart, "Worm Propagation and Countermeasures." [Online]. Available: http://www.sans.org/reading_room/whitepapers/malicious/worm-propagation-countermeasures_1410 [Accessed: March 11, 2011].

[GLM10]   Walter Glenn, Scott Lowe and Joshua Maher, "Microsoft Exchange Server 2007 Administrator's Companion." [Online]. Available: http://technet.microsoft.com/en-us/library/cc505928.aspx [Accessed: November 18, 2010].

[Gul09]   Ron Gula, "Enchanced Operating System Identification with Nessus." [Online]. Available: http://blog.tenablesecurity.com/2009/02/enhanced_operat.html [Accessed: Februtary 7, 2011].

[HPK04]   Mark Handley, Vern Paxson and Christian Kreibich, "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics." [Online]. Available: http://www.ece.cmu.edu/~adrian/731-sp04/readings/HPK-nids.pdf [Accessed: May 15, 2010].

[KaS10]   Ratinder Kaur and Maninder Singh, "Hardening OS Identity by Customised Masking Techniques." [Online]. Available: http://amrita.edu/cyber-workshop/proceedings/icscf09_submission_67.pdf [Accessed: April 27, 2010].

[KFL01]   Dorene Kewley, Russ Fink, John Lowry and Mike Dean, "Dynamic Approaches to Thwart Adversary Intelligence Gathering." [Online]. Available: http://www.bbn.com/resources/pdf/DISCEX_DYNAT.pdf

[Accessed: March 16, 2010].

[Kol05]   Eric Kollmann, "Chatter on the Wire: A look at excessive network traffic and what it can mean to network security." [Online]. Available: http://myweb.cableone.net/xnih/download/OS%20FingerPrint.pdf [Accessed: May 19, 2010].

[Kol07]   Eric Kollmann, "Chatter on the Wire: A look at DHCP traffic." [Online]. Avaliable: http://myweb.cableone.net/xnih/download/chatter-dhcp.pdf [Accessed: May 19, 2010].

[LeT04]   Henry C.J. Lee and Vrizlynn L.L. Thing, "Port Hopping for Resilient Networks." [Online]. Available: http://icsd.i2r.a-star.edu.sg/publications/HentyLee_2004_VTC_0406_15.pdf [Accessed: February 5, 2010].

[LiT08]   Janne Lindqvist and Juha-Matti Tapio, "Protecting Privacy with Protocol Stack Virtualization," in Proceedings of the 7th ACM workshop on Privacy in the electronic society, pp. 65-74, 2008.

[MaB10]   David Maltz and Pravin Bhagwat, "Improving HTTP Caching Proxy Performance with TCP Tap." [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.6843&rep=rep1&type=pdf [Accessed: May 17, 2010].

[Mic05]   Microsoft Corporation, "Exchange Server 2003." [Online]. Available: http://technet.microsoft.com/en-us/library/bb123872(EXCHG.65).aspx [Accessed: January 21, 2011].

[Mic09]   Microsoft Corporation, "Installing Windows SharePoint Services 3.0 on a Server Running Windows Small Business Server 2003." [Online]. Available: http://technet.microsoft.com/en-us/library/cc671966(WS.10).aspx [Accessed: January 31, 2011].

[MPS02]   John Michalski, Carrie Price, Eric Stanton and Erik Lee, "Network Security Mechanisms Utilizing Dynamic Network Address Translation." [Online]. Available: http://prod.sandia.gov/techlib/access-control.cgi/2002/023613.pdf [Accessed: February 8, 2010].

[Mur09]   Sherry B. Murphy, "Deceiving Adversary Network Scanning Efforts Using Host-Based Deception." [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA502233 [Accessed: April 16, 2010].

[Naz04]   Jose Nazario, *Defense and Detection Strategies against Internet Worms*. Artech House, 2004.

[Net09]   Netcraft, "January 2009 Web Server Survey." [Online]. Available: http://news.netcraft.com/archives/2009/01/16/january_2009_web_server_survey.html [Accessed: July 23, 2010].

[Osr03]   "Network Architecture for Kernel-Mode Driver." [Online]. Available: http://www.osronline.com/ddkx/network/102gen_3kvb.htm [Accessed: February 18, 2011].

[PaF01]   Jitendra Pahdye and Sally Floyd, "On inferring TCP behavior," in Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communication, pp. 287-298, 2001.

[Rep08]   Keith A. Repik, "Defeating Adversary Network Intelligence Efforts With Active Cyber Defense Techniques," in Air University Research Information System. [Online]. Available: http://wf2dnvr6.webfeat.org/K9XHO15726/url=https://www.afresearch.org/skins/rims/display.aspx?moduleid=be0e99f3-fc56-4ccb-8dfe-670c0822a153&mode=user&action=downloadpaper&objectid=b03aa817-150e-442a-9d38-a2bdf21aabf8&rs=PublishedSearch [Accessed: March 14, 2010].

[Rob06]   Drew Robb, "Hardware Today: Server Virtualization Becoming Norm." [Online]. Available: http://www.serverwatch.com/hreviews/article.php/3639556/Hardware-Today-Server-Virtualization-Becoming--Norm.htm [Accessed: July 23, 2010].

[RoS01]   Gael Roualland and Jean-Marc Saffroy, "IP Personality." [Online]. Available: http://ippersonality.sourceforge.net/doc/ippersonality-en.html [Accessed: April 20, 2010].

[Sha04]   Saumil Shah, "An Introduction to HTTP fingerprinting." [Online]. Available: http://net-square.com/httprint/httprint_paper.html [Accessed: May 17, 2010].

[SiB07]   Ken Simpson and Stas Bekman, "Fingerprinting the World's Mail Servers." [Online]. Available: http://oreilly.com/pub/a/sysadmin/2007/01/05/fingerprinting-mail-

servers.html?page=last [Accessed: June 4, 2010].

[SkL08]   Ed Skoudis and Tom Liston, *Counter Hack Reloaded*, 2nd ed. Stoughton, MA: Pearson Education, 2008.

[SMJ00]   Matthew Smart, G. Robert Malan and Farnam Jahanian, "Defeating TCP/IP Stack Fingerprinting," in 9th USENIX Security Symposium Paper 2000, pp. 229-240, 2000.

[Spa03]   Ryan Spangler, "Analysis of Remote Active Operating System Fingerprinting Tools," May 2003. [Online]. Available: http://www.packetwatch.net/documents/papers/osdetection.pdf [Accessed: April 20, 2010].

[Spo05]   Mark Sportack, "The ABCs of TCP/IP," March 2005. [Online.] Available: http://www.ciscopress.com/articles/article.asp?p=377101 [Accessed: March 4, 2011].

[Tal03]   Greg Taleck, "Ambiguity Resolution via Passive OS Fingerprinting." [Online]. Available: http://www.itsec.gov.cn/webportal/download/2003-Ambiguity%2520Resolution%2520via%2520Passive%2520OS%2520Fingerprinting [Accessed: May 19, 2010].

[Tor05]   Politecnico di Torino, "Winpcap Internals." [Online]. Available: http://www.winpcap.org/docs/docs_412/html/group__internals.html [Accessed: February 7, 2011].

[W3s11]   "OS Platform Statistics." [Online]. Available: http://www.w3schools.com/browsers/browsers_os.asp [Accessed: February 18, 2011].

[Wei11]   Eric W. Weisstein, "Fisher's Exact Test." From MathWorld—A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/FishersExactTest.html [Accessed: March 16, 2011].

[WJS07]   Haining Wang, Cheng Ji and Kang G. Shin, "Defense against spoofed IP traffic using hop-count filtering," in IEEE/ACM Transactions on Networking (TON), vol. 15, no. 1, pp. 40-53, 2007.

[Wol02]   Mark Wolfgang, "Host Discovery with Nmap." [Online]. Available: http://www.rootsecure.net/content/downloads/pdf/nmap_host_discovery.pdf [Accessed: May 24, 2010].

[WSM04] David Watson, Matthew Smart, G. Robert Malan and Farnam Jahanian, "Protocol Scrubbing: Network Security Through Transparent Flow Modification," in Networking, IEEE/ACM Transactions, vol. 12, no. 2, pp. 261-273, 2004.

[Yua05] Li Yuan, "Companies Face System Attacks From Inside Too," in The Wall Street Journal, pp. B1, June 1, 2005.

[Yui06] James Joseph Yuill, "Defensive Computer-Security Deception Operations: Processes, Principles and Techniques." [Online]. Available: http://www.lib.ncsu.edu/theses/available/etd-10272006-055733/unrestricted/etd.pdf [Accessed: May 19, 2010].

[VCH02] Franck Veysset, Olivier Courtay and Olivier Heen, "New Tool And Technique For Remote Operating System Fingerprinting," April 2002. [Online]. Available: http://www.ouah.org/ring-full-paper.pdf [Accessed: May 17, 2010].

1. REPORT DATE *(DD-MM-YYYY)* 16-06-2011	2. REPORT TYPE Master's Thesis	3. DATES COVERED *(From – To)* June 2009 – June 2011
4. TITLE AND SUBTITLE Host-Based Systemic Network Obfuscation System for Windows		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Huber, Kevin E.		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765		**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GCO/ENG/11-05
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) n/a		**10. SPONSOR/MONITOR'S ACRONYM(S)** n/a
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Network traffic identifies the operating system and services of the host that created the traffic. Current obfuscation programs focus solely on the Transport and Internet layer protocols of the TCP/IP model. Few obfuscation programs were developed to run on a Windows operating system to provide host-based obfuscation. Systemic Network Obfuscation System (SNOS) was developed to provide a thorough obfuscation process for network traffic on the Windows operating system. SNOS modifies the protocols found at all layers of the TCP/IP model to effectively obfuscate the Windows operating system and services running on the host.

15. SUBJECT TERMS
Obfuscation, OS Fingerprinting, TCP/IP Fingerprinting, Nmap, Nessus, NDIS Intermediate driver

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Dr. Barry Mullins (ENG)
REPORT U	**ABSTRACT** U	**c. THIS PAGE** U	UU	186	19b. TELEPHONE NUMBER *(Include area code)* (937) 785-3636, x7979 barry.mullins@afit.edu

Standard Form 298 (Rev: 8-98)
Prescribed by ANSI Std. Z39-18